UNIVERSITY
OF TRENTO

DIPARTIMENTO DI INGEGNERIA E SCIENZA DELL'INFORMAZIONE

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.disi.unitn.it

A LOGIC FOR CONTRACTS

Massimo Bartoletti and Roberto Zunino

June 2009 (submitted), November 2009 (revised)

Technical Report #DISI-09-034

A Logic for Contracts

Massimo Bartoletti
Dipartimento di Matematica e Informatica
Università degli Studi di Cagliari, Italy

Roberto Zunino
Dipartimento di Ingegneria e Scienza dell'Informazione
Università degli Studi di Trento, Italy

Abstract

We investigate the logical foundations of *contracts* in distributed applications. A contract is an agreement stipulated between two or more parties, which specifies the duties and the rights of the parties involved therein. We model contracts as formulae in an intuitionistic logic extended with a "contractual" form of implication $\twoheadrightarrow$. This supports for a variant of Modus Ponens, where from a "promise" $a \twoheadrightarrow b$ to deduce b, one does not need to know a; yet, it suffices to have a dual promise $b \twoheadrightarrow a$. We study the proof theory for our logic. In particular, we provide it with a Hilbert-style axiomatisation, which is shown consistent, and with a Gentzen-style sequent calculus, shown equivalent to the axiomatization. We prove our logic decidable, via a cut elimination property. The rights and the duties deriving from any set of contracts can therefore be mechanically inferred.

1 Introduction

Security, trustworthiness and reliability of software systems are crucial issues in the rising Information Society. As new online services (e-commerce, e-banking, e-government, etc.) are made available, the number and the criticality of the problems related to non-functional properties of services keeps growing. From the client point of view, it is important to be sure that, e.g., after a payment has been made, then either the payed goods are made available, or a full refund is issued. From the provider point of view it is important to be sure, e.g., that a client will not repudiate a completed transaction, so to obtain for free the goods already delivered. In other words, the interaction between a client and a service must be regulated by a suitable contract, which guarantees to both parties the properties on demand. The crucial problem is then how to define the concept of *contract*, and how to actually enforce it, in an environment - the Internet - which is by definition open and unreliable.

Unfortunately, at the present no widespread technology seems to give a general solution to this problem. Typically, services do not provide the client with any concrete guarantee about the actual functionality they implement. At best, the service provider commits himself to respect some "service level agreement".

In the case this is not honoured, the only thing the client can do is to take legal steps against the provider (or *vice versa*). Although this is the normal practice nowadays, it is highly desirable to reverse this trend. Indeed, both clients and services could incur relevant expenses due to the needed legal disputes. This is impractical, especially for transactions dealing with small amounts of money.

Contributions. We study the theoretical foundations upon which constructing a service infrastructure where contracts carry, besides the usual legal meaning, also a "formal" one. In other words, our contracts will be mathematical entities, that specify exactly the rights and the duties of clients and services. We envisage a world of services where clients and service providers can have precise, mathematical guarantees about the implemented features, and about the assumed side conditions. In the scenario we aim at, contesting a contract will not necessarily require to resort to a court, yet it will be an event managed automatically, deterministically and inexpensively, by the service infrastructure itself. We call this interaction paradigm "contract-based computing".

In this paper we begin our investigation on contract-based computing, by studying formalisms to describe contracts, and to reason about them. A contract is a binding agreement between two or more parties, that dictates the duties the involved parties must fulfill, whenever some preconditions are satisfied. Our theory of contracts will be able to infer, in each possible context, the duties deriving from a given set of contracts. To put the developed theory at work, we have implemented a proof search tool, which decides whether a given formula is a tautology or not [26].

Summary. The paper is organized as follows:

- In Section 2 we give, with the help of an example, some motivations about the need for a logic for contracts.
- In Section 3 we devise a minimal set of properties which are desirable in any logical formalization of contracts.
- In Section 4 we define our logic for contracts, through a Hilbert-style axiomatization. Our logic satisfies all the properties identified above as desirable.
- In Section 5 we give further details and examples about using our logic to model a variety of contracts.
- In Section 6 we provide our logic with a Gentzen-style sequent calculus, which is equivalent to the Hilbert-style axiomatization.
- In Section 7 we prove the main technical result about our logic, that is its decidability. This is obtained by showing that our sequent calculus enjoys cut elimination and the subformula property.
- In Section 8 we study relations with other logics, in particular with intuitionistic propositional logic IPC, with the modal logic S4, and with propositional lax logic.

- In Section 9 we extend our logic with an indexed modality, to model principals. Contract agreements then come in a richer flavour, because of the binding between the contracting parties and their inferred duties, which is now revealed.
- In Section 10 we discuss some related work.
- In Section 11 we conclude, by discussing some possible future work and extensions to our logic.

2 Motivations

Suppose there are two kids, Alice and Bob, who want to play together. Alice has a toy airplane, while Bob has a bike. Both Alice and Bob wish to play with each other's toy: Alice wants to ride Bob's bike, and Bob wants to play with Alice's airplane. Alice and Bob are very meticulous kids, so before sharing their toys, they stipulate the following "gentlemen's agreement":

Alice: I will lend my airplane to you, Bob, provided that I borrow your bike.

Bob: I will lend my bike to you, Alice, provided that I borrow your airplane.

We want to make precise the commitments exchanged by Alice and Bob, so to be able to formally deduce that Alice and Bob will indeed share their toys, provided they are real "gentlemen" who always respect their promises.

Let us write a for the atomic proposition "Alice lends her airplane" and b for "Bob lends his bike". Using classical propositional logic, a straightforward – yet naïve – formalization of the above commitments could be the following. Alice's commitment A is represented as the formula $\mathsf{b} \to \mathsf{a}$ (if Bob lends his bike, then Alice lends her airplane) and Bob's commitment as the formula $\mathsf{a} \to \mathsf{b}$ (if Alice lends her airplane, then Bob lends his bike):

$$A \;=\; \mathsf{b} \to \mathsf{a} \qquad\qquad B \;=\; \mathsf{a} \to \mathsf{b}$$

where the symbol $\to$ denotes classical implication. Under the hypothesis that Alice and Bob always respect their promises, both formulas A and B are *sound* with respect to our scenario. For the formula A, it is true that whenever b holds (Bob lends his bike), then a will also hold (Alice lends her airplane). For the formula B, it is true that whenever a holds (Alice lends her airplane), then b will also hold (Bob lends his bike).

So, why are we unhappy with the above formalization? The problem is that, in classical propositional logic, the above commitments are not enough to deduce that Alice will lend her airplane and Bob will lend his bike. Formally, it is possible to make true the formula $A \wedge B$ by assigning false to both propositions a and b. So, Alice and Bob will not be able to play together, despite of their gentlemen's agreement, and of the hypothesis that they always respect promises.

The failure to represent scenarios like the one above seems related to the "standard" interpretation of the Modus Ponens. In both classical and intuitionistic proof theories, the Modus Ponens rule allows to deduce b whenever $\mathsf{a} \to \mathsf{b}$ and a are true. Back to our scenario, we could deduce that Bob lends his bike, but only *after* Alice has lent Bob her airplane. One of the two parties must

"take the first step" in order to make the agreement become effective, that is to imply the promised duties. In a logic for mutual agreements, we would like – instead – to make an agreement become effective also without the need of some party taking the first step (as we shall see in a while, such party might not even exists in more complex scenarios). That is, A and B are *contracts*, that once stipulated imply the duties promised by all the involved parties.

Technically, we would like our logic able to deduce $\mathsf{a} \wedge \mathsf{b}$ whenever $A \wedge B$ is true. As we have noticed above, this does not hold neither in classical nor in intuitionistic propositional logic, where the behaviour of implication strictly adheres to Modus Ponens.

To model contracts, we extend the intuitionistic propositional logic IPC [30] with a new form of implication, that we denote with the symbol $\twoheadrightarrow$. The resulting logic is called PCL, for Propositional Contract Logic. For instance, the contract declared by Alice, "I will lend my airplane to Bob provided that Bob lends his bike to me", will be written $\mathsf{b} \twoheadrightarrow \mathsf{a}$. This form of, say, *contractual* implication, is stronger than the standard implication $\rightarrow$ of IPC. Actually, $\mathsf{b} \twoheadrightarrow \mathsf{a}$ implies a not only when b is true, like IPC implication, but also in the case that a "compatible" contract, e.g. $\mathsf{a} \twoheadrightarrow \mathsf{b}$, holds. In our scenario, this means that Alice will lend her airplane to Bob, provided that Bob agrees to lend his bike to Alice whenever he borrows Alice's airplane, and *vice versa*. Actually, the following formula is a theorem of our logic:

$$(\mathsf{b} \twoheadrightarrow \mathsf{a}) \ \wedge \ (\mathsf{a} \twoheadrightarrow \mathsf{b}) \ \rightarrow \ \mathsf{a} \wedge \mathsf{b}$$

In other words, from the "gentlemen's agreement" stipulated by Alice and Bob, we can deduce that the two kids will indeed share their toys.

To make our scenario a bit more interesting, suppose now that a third kid, Carl, joins Alice and Bob. Carl has a comic book, which he would share with Alice and Bob, provided that he can play with the other kids' toys. To accommodate their commitments to the new scenario, the three kids decide to stipulate the following gentlemen's agreement (which supersedes the old one):

Alice: I will share my airplane, provided that I can play with Bob's bike and read Carl's comic book.

Bob: I will share my bike, provided that I can play with Alice's airplane and read Carl's comic book.

Carl: I will share my comic book, provided that I can play with Alice's airplane and ride Bob's bike.

Let us write a for "Alice shares her airplane", b for "Bob shares his bike", and c for "Carl shares his comic book". Then, the above commitments can be rephrased as: Alice promises a provided that b and c, Bob promises b provided that a and c, and Carl promises c provided that a and b. In our contract logic, we model the above agreement as the formula $A \wedge B \wedge C$, where:

$$A \ = \ (\mathsf{b} \wedge \mathsf{c}) \twoheadrightarrow \mathsf{a} \qquad\qquad B \ = \ (\mathsf{a} \wedge \mathsf{c}) \twoheadrightarrow \mathsf{b} \qquad\qquad C \ = \ (\mathsf{a} \wedge \mathsf{b}) \twoheadrightarrow \mathsf{c}$$

The proof system of our logic will be able to deduce that the three kids will indeed share their toys, that is, the following is theorem of the logic:

$$A \wedge B \wedge C \ \ \rightarrow \ \ \mathsf{a} \wedge \mathsf{b} \wedge \mathsf{c}$$

It it interesting to compare the specification above, which uses contractual implication, with a specification which uses, instead, standard implication. Let:

$$A' \;=\; (\mathsf{b} \wedge \mathsf{c}) \rightarrow \mathsf{a} \qquad\qquad B' \;=\; (\mathsf{a} \wedge \mathsf{c}) \rightarrow \mathsf{b} \qquad\qquad C' \;=\; (\mathsf{a} \wedge \mathsf{b}) \rightarrow \mathsf{c}$$

Clearly, in this case we cannot deduce $A' \wedge B' \wedge C' \rightarrow \mathsf{a} \wedge \mathsf{b} \wedge \mathsf{c}$. This scenario provide us with a further insight about contractual implication. Reconsider for a moment the scenario with only two contracting parties, modelled with standard implication: $(\mathsf{a} \rightarrow \mathsf{b}) \wedge (\mathsf{b} \rightarrow \mathsf{a})$. We have shown above that, in such a situation, a single party can make the agreement effective, e.g. Alice can take the first step, and lend her airplane to Bob (then, by Modus Ponens, Bob will lend his bike to Alice). Instead, in the extended scenario (modelled with standard implication), it is no longer the case that a single party can take the first step and achieve the same goal. For instance, assume that Alice decides unilaterally to share her airplane. Even by doing that, Alice will have no guarantee that, eventually, she will be able to play with the other kids' toys. This is because, with standard implication, setting a to true would transform $A' \wedge B' \wedge C'$ into $(\mathsf{c} \rightarrow \mathsf{b}) \wedge (\mathsf{b} \rightarrow \mathsf{c})$, which clearly implies neither b nor c. To make their agreement effective, at least two of the three parties must use contractual implication in their commitments (while the other one can use standard implication).

This observation can be generalised to a scenario with n contracting parties, where one can show that at least $n-1$ parties must use contractual implication.

3 Desirable properties

We now discuss some desirable properties of a logic for contracts, as well as some other properties that – instead – are undesirable. In the next section, we will show an axiomatisation that enjoys all the properties marked here as desired.

As shown in the previous section, a characterizing property of contractual implication is that of allowing two contracting parties to "handshake", so to make their agreement effective. This is resumed by the following *handshaking* property, which we expect to hold for any logic for contracts:

$$(p \twoheadrightarrow q) \wedge (q \twoheadrightarrow p) \;\rightarrow\; p \wedge q \tag{1}$$

A generalisation of the above property to the case of n contracting parties is also desirable. It is a sort of "circular" handshaking, where the $(i+1)$-th party, in order to promise some duty p_{i+1}, relies on a promise p_i made by the i-th party. In the case of n parties, we would expect the following:

$$(p_1 \twoheadrightarrow p_2) \wedge \cdots \wedge (p_{n-1} \twoheadrightarrow p_n) \wedge (p_n \twoheadrightarrow p_1) \;\rightarrow\; p_1 \wedge \cdots \wedge p_n \tag{2}$$

As a concrete example, consider an e-commerce scenario where a Buyer can buy items from a Seller, and pay them through a credit card. To mediate the interaction between the Buyer and the Seller, there is a Bank which manages payments. The contracts issued by the three parties could then be:

Buyer: I will click "pay" provided that the Seller will ship my item

Seller: I will ship your item provided that I get the money

Bank: I will transfer money to the Seller provided that the Buyer clicks "pay".

Let the atomic propositions ship, $\mathsf{clickpay}$, and pay denote respectively the facts "Seller ships item", "Buyer clicks pay", and "Bank transfers money". Then, the three contracts above can be modelled as follows:

$$\mathit{Buyer} = \mathsf{ship} \twoheadrightarrow \mathsf{clickpay} \qquad \mathit{Bank} = \mathsf{clickpay} \twoheadrightarrow \mathsf{pay} \qquad \mathit{Seller} = \mathsf{pay} \twoheadrightarrow \mathsf{ship}$$

Then, by the handshaking property (2), we obtain a successful transaction:

$$\mathit{Buyer} \wedge \mathit{Bank} \wedge \mathit{Seller} \;\rightarrow\; \mathsf{pay} \wedge \mathsf{ship}$$

Note that, in the special case that n equals 1, the above "circular" handshaking property turns into a particularly simple form:

$$(p \twoheadrightarrow p) \;\rightarrow\; p \tag{3}$$

Intuitively, (3) can be interpreted as the fact that promising p provided that p, implies p (actually, also the converse holds, so that promise is equivalent to p).

A generalisation of the scenario of the previous section to the case of n kids is also desirable. It is a sort of "greedy" handshaking property, because now a party promises p_i only provided that *all* the other parties promise their duties, i.e. $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_n$. The greedy handshaking can then be stated as:

$$\bigwedge_{i \in 1..n} \Big((p_1 \wedge \ldots \wedge p_{i-1} \wedge p_{i+1} \wedge \ldots \wedge p_n) \twoheadrightarrow p_i \Big) \;\rightarrow\; p_1 \wedge \cdots \wedge p_n \tag{4}$$

As shown by (1), a contract $p \twoheadrightarrow q$ becomes effective, i.e. implies the promise q, when it is matched by a dual contract $q \twoheadrightarrow p$. Even more directly, $p \twoheadrightarrow q$ should be effective also in the case that the premise p is already true:

$$p \wedge (p \twoheadrightarrow q) \;\rightarrow\; q \tag{5}$$

In other words, contractual implication should be *stronger* than standard implication, i.e. we expect that the following is a theorem of any logic for contracts:

$$(p \twoheadrightarrow q) \rightarrow (p \rightarrow q) \tag{6}$$

On the other hand, we do not want that also the converse holds, since this would equate the two forms of implication:

$$(p \rightarrow q) \rightarrow (p \twoheadrightarrow q) \qquad \text{NOT A TAUTOLOGY}$$

We want contractual implication to share with standard implication a number properties. We discuss some of them below. First, a contract that promises true (written $\top$) is always satisfied, regardless of the precondition. Then, we expect the following tautology:

$$p \twoheadrightarrow \top \tag{7}$$

However, differently from standard implication, we do not want that a contract with a false precondition (written $\bot$) always holds.

$$\bot \twoheadrightarrow p \qquad \text{NOT A TAUTOLOGY}$$

So see why, assume that we have $\bot \twoheadrightarrow p$ as a tautology, for all p. Then, it would also be the case for $p = \bot$, and so by the binary handshaking property we would deduce a contradiction: $(\bot \twoheadrightarrow \bot) \wedge (\bot \twoheadrightarrow \bot) \rightarrow \bot$.

Another property of implication that we want to preserve is transitivity:

$$(p \twoheadrightarrow q) \ \wedge \ (q \twoheadrightarrow r) \ \rightarrow \ (p \twoheadrightarrow r) \tag{8}$$

Back to our previous example, transitivity would allow the promise of the Buyer ($\mathsf{ship} \twoheadrightarrow \mathsf{clickpay}$) and the promise of the Bank ($\mathsf{clickpay} \twoheadrightarrow \mathsf{pay}$) to be combined in the promise $\mathsf{ship} \twoheadrightarrow \mathsf{pay}$.

Contractual implication should also enjoy a stronger form of transitivity. We illustrate it with the help of an example. Suppose an air-flight customer wants to book a flight. To do that, he issues the following contract:

$$\mathit{Customer} : \mathsf{bookFlight} \twoheadrightarrow \mathsf{pay}$$

The contract states that the customer promises to pay the required amount, provided that he obtains a flight reservation. Suppose now that an airline company starts a special offer, in the form of a free drink for each customer that makes a reservation:

$$\mathit{AirLine} : \mathsf{pay} \twoheadrightarrow \mathsf{bookFlight} \wedge \mathsf{freeDrink}$$

Of course, the two contracts should give rise to an agreement, because the airline company is promising a better service than the one required be the customer contract. To achieve that, we expect to be able to "weaken" the contract of the airline company, to make it match the contract issued by the customer:

$$\mathit{AirLine} \ \rightarrow \ (\mathsf{pay} \twoheadrightarrow \mathsf{bookFlight})$$

Alternatively, one could make the two contracts match by making stronger the precondition required by the customer, that is:

$$\mathit{Customer} \ \rightarrow \ (\mathsf{bookFlight} \ \wedge \ \mathsf{freeDrink} \twoheadrightarrow \mathsf{pay})$$

More in general, we want the following two properties hold for any logic for contracts. They say that the promise in a contract can be arbitrarily weakened (9), while the precondition can be arbitrarily strengthened (10).

$$(p \twoheadrightarrow q) \ \wedge \ (q \rightarrow q') \ \rightarrow \ (p \twoheadrightarrow q') \tag{9}$$

$$(p' \rightarrow p) \ \wedge \ (p \twoheadrightarrow q) \ \rightarrow \ (p' \twoheadrightarrow q) \tag{10}$$

Note that the properties (8), (9) and (10) cover three of the four possible cases of transitivity properties which mix standard and contractual implication. Observe, instead, that combining two implications into a contract is *not* a desirable property of any logic for contracts, for the same reason for which we do not want standard and contractual implications be equivalent.

$$(p \rightarrow q) \ \wedge \ (q \rightarrow r) \ \rightarrow \ (p \twoheadrightarrow r) \qquad \text{NOT A TAUTOLOGY}$$

Another property that should hold is that, if a promise q is already true, then it is also true any contract which promises q:

$$q \ \rightarrow \ (p \twoheadrightarrow q) \tag{11}$$

Of course, we do not want the converse to hold: it is not always the case that a contract implies its promise.

$$(p \twoheadrightarrow q) \rightarrow q \qquad \text{NOT A TAUTOLOGY}$$

4 A Logic for Contracts

In this section we give the basic ingredients of our logic for contracts PCL. In Sect. 4.1 we present the syntax of PCL; in Sect. 4.2 we provide it with an Hilbert-style axiomatization. Finally, in Sect. 4.3 we study some interesting properties of PCL that follow from the given axioms.

4.1 Syntax

The syntax of PCL is a simple extension of IPC. It includes the standard logic connectives $\neg, \wedge, \vee, \rightarrow$ and the contractual implication connective $\twoheadrightarrow$. We assume a denumerable set $\{\mathsf{p}, \mathsf{q}, \mathsf{r}, \mathsf{s}, \ldots\}$ of prime (atomic) formulae. Arbitrary PCL formulae are denoted with the letters $p, q, r, s, \ldots$ (note that the font differs from that used for prime formulae). The precedence of IPC operators is the following, from highest to lowest: $\neg, \wedge, \vee, \rightarrow$. We stipulate that $\twoheadrightarrow$ has the same precedence as $\rightarrow$.

Definition 4.1. The formulae of PCL are inductively defined by the following grammar.

$p ::=$	$\bot$	false
$\mid$	$\top$	true
$\mid$	p	prime
$\mid$	$\neg p$	negation
$\mid$	$p \vee p$	disjunction
$\mid$	$p \wedge p$	conjunction
$\mid$	$p \rightarrow p$	implication
$\mid$	$p \twoheadrightarrow p$	contractual implication

We let $p \leftrightarrow q$ be syntactic sugar for $(p \rightarrow q) \wedge (q \rightarrow p)$. If a formula is $\twoheadrightarrow$-free, we say it is an IPC formula. We use the symbol $\Longrightarrow$ to denote implication in the meta-theory, so to avoid confusion with $\rightarrow$.

4.2 Proof Theory: Hilbert-style Axiomatization

We now define a Hilbert-style proof system for PCL. The axioms include all the standard axioms for IPC (see e.g. [24]). Like in IPC, we have a single inference rule, i.e. Modus Ponens. The characterising axioms for PCL are called Zero, Fix and PrePost.

Definition 4.2. The axioms of PCL are presented below.

- Core IPC axioms.

$p \wedge q \to p$	$\wedge 1$
$p \wedge q \to q$	$\wedge 2$
$p \to q \to p \wedge q$	$\wedge 3$
$p \to p \vee q$	$\vee 1$
$q \to p \vee q$	$\vee 2$
$(p \to r) \to (q \to r) \to (p \vee q) \to r$	$\vee 3$
$p \to q \to p$	$\to 1$
$(p \to q \to r) \to (p \to q) \to p \to r$	$\to 2$
$\bot \to p$	$\bot$
$\top$	$\top$
$\neg p \to p \to q$	$\neg 1$
$(p \to q) \to (p \to \neg q) \to \neg p$	$\neg 2$

- Contractual implication axioms.

$\top \twoheadrightarrow \top$	Zero
$(p \twoheadrightarrow p) \to p$	Fix
$(p' \to p) \to (p \twoheadrightarrow q) \to (q \to q') \to (p' \twoheadrightarrow q')$	PrePost

- *Modus ponens* (cut).

$$\frac{p \quad p \to q}{q}\ \mathsf{Cut}$$

We write $\vdash p$ when p is derivable from the axioms and inference rules above.

The contractual implication axioms are actually special cases of the desired properties of contracts seen in Sect. 3. For instance, the axiom Zero is a special case of (7). The axiom Fix is exactly the property (3). The axiom PrePost plays the role of both the properties (10) and (9).

4.3 Fundamental Consequences

We present below some significant consequences of the axioms in Def. 4.2. Note that these consequences cover all the properties marked in Sect. 3 as desirable. To shorten our notation, when speaking about non-provability, we write $\nvdash p$ when the formula p is not a theorem of PCL (i.e. p is not true for all the instantiations of the metavariables). For instance, we write $\nvdash p \to p \wedge q$ to mean that $\neg \forall p, q.\ \vdash p \to p \wedge q$.

Lemma 4.3. *Contractual implication is strictly stronger than implication.*

$$\vdash (p \twoheadrightarrow q) \to (p \to q) \tag{12}$$
$$\nvdash (p \to q) \to (p \twoheadrightarrow q) \tag{13}$$

Proof. For (12), assume that $p \twoheadrightarrow q$ and p hold. Hence, $q \to p$ trivially holds. By using PrePost on $q \to p$,$p \twoheadrightarrow q$, and $q \to q$, we get $q \twoheadrightarrow q$. We then conclude q by Fix. We anticipate in (13) a negative result that can be mechanically verified using the decision procedure of Lemma 6.14. □

Below, we establish some further connections between $\twoheadrightarrow$ and $\to$, which generalize the transitivity of $\twoheadrightarrow$.

Lemma 4.4. *Contractual implication is transitive. More in detail, we have the following interactions between $\twoheadrightarrow$ and $\rightarrow$.*

$$\vdash (p \twoheadrightarrow q) \rightarrow (q \twoheadrightarrow r) \rightarrow (p \twoheadrightarrow r) \tag{14}$$
$$\vdash (p \rightarrow q) \rightarrow (q \twoheadrightarrow r) \rightarrow (p \twoheadrightarrow r) \tag{15}$$
$$\vdash (p \twoheadrightarrow q) \rightarrow (q \rightarrow r) \rightarrow (p \twoheadrightarrow r) \tag{16}$$
$$\nvdash (p \rightarrow q) \rightarrow (q \rightarrow r) \rightarrow (p \twoheadrightarrow r) \tag{17}$$

Proof. Properties (15,16) are direct consequences of PrePost, and the trivial $r \rightarrow r$ and $p \rightarrow p$.

For (14), we apply Lemma 4.3(12) to $p \twoheadrightarrow q$, so obtaining $p \rightarrow q$. Then we can apply (15) to conclude.

We anticipate in (17) a negative result that can be mechanically verified using the decision procedure of Lemma 6.14. □

The distributivity laws of $\twoheadrightarrow$ are quite peculiar. As for standard implication in IPC, $\vee$-distributivity holds in only one direction (18,19). Instead, while $\wedge$-distributivity holds in both directions in IPC for standard implication, contractual implication only satisfies one direction (20,21). However, a related property holds (22).

Lemma 4.5. *Distributivity laws.*

$$\vdash (p \twoheadrightarrow q) \vee (p \twoheadrightarrow r) \rightarrow (p \twoheadrightarrow (q \vee r)) \tag{18}$$
$$\nvdash (p \twoheadrightarrow (q \vee r)) \rightarrow (p \twoheadrightarrow q) \vee (p \twoheadrightarrow r) \tag{19}$$
$$\vdash (p \twoheadrightarrow (q \wedge r)) \rightarrow (p \twoheadrightarrow q) \wedge (p \twoheadrightarrow r) \tag{20}$$
$$\nvdash (p \twoheadrightarrow q) \wedge (p \twoheadrightarrow r) \rightarrow (p \twoheadrightarrow (q \wedge r)) \tag{21}$$
$$\vdash (p \twoheadrightarrow q) \wedge (q \twoheadrightarrow r) \rightarrow (p \twoheadrightarrow (q \wedge r)) \tag{22}$$

Proof. For (18), assume $(p \twoheadrightarrow q) \vee (p \twoheadrightarrow r)$. If $p \twoheadrightarrow q$ holds, we apply PrePost to weaken q to $q \vee r$. The $p \twoheadrightarrow r$ case is similar.

For (20), assume $p \twoheadrightarrow (q \wedge r)$. By PrePost, it is then easy to obtain both $p \twoheadrightarrow q$ and $p \twoheadrightarrow r$.

We anticipate in (19,21) some negative results that can be mechanically verified using the decision procedure of Lemma 6.14.

For (22), assume the hypotheses. We apply Lemma 4.3 to $q \twoheadrightarrow r$ and obtain $q \rightarrow r$, hence $q \rightarrow (q \wedge r)$. By PrePost on $p \twoheadrightarrow q$, we obtain the thesis. □

Lemma 4.6. *Substitution of equivalent formulae.*

$$\vdash (p \leftrightarrow p') \rightarrow (q \leftrightarrow q') \rightarrow (p \twoheadrightarrow q) \rightarrow (p' \twoheadrightarrow q')$$

Proof. Apply PrePost to $p' \rightarrow p$, $p \twoheadrightarrow q$, and $q \rightarrow q'$. □

The following lemma states a sufficient condition and a necessary condition for $p \twoheadrightarrow q$ to hold. These conditions are expressed in IPC, i.e. they make no use of $\twoheadrightarrow$. We will return on these in Def. 8.6, 8.8 and related results, where we will prove that, when p, q are IPC formulae, these conditions are actually the weakest sufficient condition and the strongest necessary condition that can be expressed within IPC (Lemma 8.10).

Lemma 4.7. *Contractual implication admits the following sufficient condition and necessary condition.*

$$\vdash q \to (p \twoheadrightarrow q) \tag{23}$$
$$\vdash (p \twoheadrightarrow q) \to ((q \to p) \to q) \tag{24}$$

Proof. For (23), assume q. We conclude by PrePost on $p \to \top$ (trivial), $\top \twoheadrightarrow \top$ (by Zero), $\top \to q$ (by q).

For (24), assume $p \twoheadrightarrow q$ and $q \to p$. By PrePost, we have $q \twoheadrightarrow q$, so we conclude by Fix. □

The following lemma justifies our choice of IPC, rather than classical logic CPC, as the basis for our logic: indeed, choosing CPC would make our contractual implication much less interesting.

Lemma 4.8. *Denote with $\vdash_C p$ the provability of p in the system of Def. 4.2 augmented with the axiom of excluded middle $(p \vee \neg p)$. Then, we have:*

$$\vdash_C (p \twoheadrightarrow q) \leftrightarrow q$$

Proof. Direct consequence of Lemma 4.7, since $((q \to p) \to q) \to q$ is a tautology of CPC (actually, it is the well-known Peirce's law). □

The following two lemmata are about "handshaking" of n contracting parties. Lemma 4.9 speaks about "circular" handshaking where the i-th party relies on the promise made by the $i-1$-th party, as in (2). Lemma 4.10 is a stronger version of this, because it allows each party to rely on the promises made by *all* the other parties, as in (4).

Lemma 4.9. (Handshaking) *For all $n \geq 0$ and for all $p_0, \ldots, p_n$:*

$$\vdash (p_0 \twoheadrightarrow p_1) \to \cdots \to (p_{n-1} \twoheadrightarrow p_n) \to (p_n \twoheadrightarrow p_0) \to (p_0 \wedge \ldots \wedge p_n)$$

Proof. Assume all the hypotheses. By repeated application of Lemma 4.4, we have $p_i \twoheadrightarrow p_i$ for all $i \in 0..n$. We then conclude by Fix. □

Lemma 4.10. (Greedy handshaking) *For all $n \geq 0$, for all $p_0, \ldots, p_n$, and for all $i, j \in 0..n$:*

$$\vdash \bigwedge_i \left(\left(\bigwedge_{j \neq i} p_j \right) \twoheadrightarrow p_i \right) \to \bigwedge_i p_i \tag{25}$$

Proof. By induction on n. The base case $n = 0$ is simple: $(\top \twoheadrightarrow p_0) \to p_0$ is proved by applying PrePost to $\top \twoheadrightarrow p_0$, hence obtaining $p_0 \twoheadrightarrow p_0$ and concluding by Fix.

In the inductive case, assume that (25) holds for $n-1$. We then prove it for n. To do that, we assume that the hypothesis $\left(\bigwedge_{j \neq i} p_j \right) \twoheadrightarrow p_i$ is true for each $i = 0..n$, and then we proceed to prove the thesis $\bigwedge_i p_i$. First, we prove the following auxiliary result:

$$p_n \to \bigwedge_{j \neq n} p_j \tag{26}$$

To prove (26), assume p_n. Then, we have $\left(\bigwedge_{j\neq i} p_j\right) \leftrightarrow \left(\bigwedge_{j\neq i, j\neq n} p_j\right)$, so by Lemma 4.6 we get $\left(\bigwedge_{j\neq i, j\neq n} p_j\right) \twoheadrightarrow p_i$ for $i = 0..n-1$. We can apply the inductive hypothesis (25), and have $\bigwedge_{j\neq n} p_j$.

Back to the inductive case, note that since i can be n, we have $\left(\bigwedge_{j\neq n} p_j\right) \twoheadrightarrow p_n$. We then conclude by (26,24). □

5 Examples

Example 5.1. The toy exchange scenario from Sect. 2 is modelled as:

$$\vdash (\mathsf{airplane} \twoheadrightarrow \mathsf{bike}) \wedge (\mathsf{bike} \twoheadrightarrow \mathsf{airplane}) \rightarrow (\mathsf{airplane} \wedge \mathsf{bike})$$

Indeed, this is a consequence of our axioms, and a special case of Lemma 4.9.

Example 5.2. We now exploit our logic to model a typical preliminary contract for a real estate sale in Italy.

Assume a buyer who is interested in buying a new house from a given seller. Before stipulating the actual purchase contract, the buyer and the seller meet to stipulate a preliminary sale contract, that fixes the terms and conditions of the purchase. Typically, this contract will indicate the price and the date when the deed of sale will take place, and it will outline the obligations for the buyer and the seller. When the preliminary contract is signed by both parties, the buyer will typically pay a part of the sale price. By the italian laws, if the seller decides not to sell the house after having signed the preliminary contract and collected the deposit, she must pay the buyer back twice the sum received. Similarly, if the buyer changes his mind and decides not to buy the house, he loses the whole deposited amount.

We model the preliminary sale contract as two PCL formulae, one for the buyer and the other for the seller. The buyer will sign the preliminary contract (signB), provided that the seller will actually sell her house (sellS), or she refunds twice the sum received ($\mathsf{refundS}$). Also, the buyer promises that if he signs the preliminary contract, than either he will pay the stipulated price (payB), or he will not pay and lose the deposit ($\mathsf{refundB}$)

$$\mathit{Buyer}: ((\mathsf{sellS} \vee \mathsf{refundS}) \twoheadrightarrow \mathsf{signB}) \wedge (\mathsf{signB} \twoheadrightarrow (\mathsf{payB} \vee (\neg\mathsf{payB} \wedge \mathsf{refundB})))$$

The seller promises that she will sign the preliminary contract (signS), provided that either the buyer promises to pay the stipulated amount, or he promises to lose the deposit. Also, the seller promises that is she signs the preliminary contract, then she will either sell her house, or will not sell and refund twice the sum received.

$$\mathit{Seller}: ((\mathsf{payB} \vee \mathsf{refundB}) \twoheadrightarrow \mathsf{signS}) \wedge (\mathsf{signS} \twoheadrightarrow (\mathsf{sellS} \vee (\neg\mathsf{sellS} \wedge \mathsf{refundS})))$$

A first consequence is that the two contracts lead to an agreement between the buyer and the seller, that is both parties will sign the preliminary contract:

$$\mathit{Buyer} \wedge \mathit{Seller} \rightarrow \mathsf{signB} \wedge \mathsf{signS} \tag{27}$$

As a second consequence, if one of the parties does not finalize the final deed of sale, than that party will refund the other:

$$Buyer \wedge Seller \wedge \neg \mathsf{payB} \rightarrow \mathsf{refundB} \tag{28}$$
$$Buyer \wedge Seller \wedge \neg \mathsf{sellS} \rightarrow \mathsf{refundS} \tag{29}$$

To prove the above, we proceed as follows. First, we apply transitivity (14) to *Buyer* and *Seller*:

$$(\mathsf{sellS} \vee \mathsf{refundS}) \twoheadrightarrow (\mathsf{payB} \vee (\neg \mathsf{payB} \wedge \mathsf{refundB}))$$
$$(\mathsf{payB} \vee \mathsf{refundB}) \twoheadrightarrow (\mathsf{sellS} \vee (\neg \mathsf{sellS} \wedge \mathsf{refundS}))$$

Then, we use PrePost:

$$(\mathsf{sellS} \vee (\neg \mathsf{sellS} \wedge \mathsf{refundS})) \twoheadrightarrow (\mathsf{payB} \vee (\neg \mathsf{payB} \wedge \mathsf{refundB}))$$
$$(\mathsf{payB} \vee (\neg \mathsf{payB} \wedge \mathsf{refundB})) \twoheadrightarrow (\mathsf{sellS} \vee (\neg \mathsf{sellS} \wedge \mathsf{refundS}))$$

So, by Lemma 4.9, we have that $Buyer \wedge Seller$ implies

$$(\mathsf{sellS} \vee (\neg \mathsf{sellS} \wedge \mathsf{refundS})) \wedge (\mathsf{payB} \vee (\neg \mathsf{payB} \wedge \mathsf{refundB}))$$

By (12) the above and $Buyer \wedge Seller$ imply $\mathsf{signB} \wedge \mathsf{signS}$. This proves (27). Also, the above and $\neg \mathsf{payB}$ clearly imply refundB. The same holds for sellS and refundS. Hence, we establish (28,29).

Example 5.3. We now describe a possible online sale between two parties. In order to buy an item, first the buyer has to contact the bank and reserve from his account a specific amount of money for the transaction. When this happens, that amount is no longer available for anything else. We model this reservation with the formula lock. Then, the buyer has to make an offer to the seller: this is modelled with offer. The seller, when provided with an offer, evaluates it. If she thinks the offer is good, and the money has been reserved, then she will send the item (send). Otherwise, she cancels the transaction (abort). When the transaction is aborted, the bank cancels the money reservation, so that the buyer can use the amount for other transactions (unlock).

We now formalize the scenario. The buyer agrees to $\mathsf{lock} \wedge \mathsf{offer}$, provided that either the item is sent, or the money reservation is cancelled. The seller agrees to evaluate the offer. The bank agrees to cancel the reservation when the transaction is aborted.

$$Buyer : (\mathsf{send} \vee \mathsf{unlock}) \twoheadrightarrow (\mathsf{lock} \wedge \mathsf{offer})$$
$$Seller : \mathsf{offer} \twoheadrightarrow ((\mathsf{lock} \rightarrow \mathsf{send}) \vee \mathsf{abort})$$
$$Bank : (\mathsf{lock} \wedge \mathsf{abort}) \twoheadrightarrow \mathsf{unlock}$$

Under these assumptions, we can see that either the item is sent, or the transaction is aborted and the reservation cancelled.

$$\vdash (Buyer \wedge Seller \wedge Bank) \rightarrow (\mathsf{send} \vee (\mathsf{abort} \wedge \mathsf{unlock}))$$

To prove this, first we apply PrePost to *Seller* and obtain $(\mathsf{lock} \wedge \mathsf{offer}) \twoheadrightarrow ((\mathsf{lock} \rightarrow \mathsf{send}) \vee \mathsf{abort})$. By property (22) and *Buyer*, we have $(\mathsf{send} \vee \mathsf{unlock}) \twoheadrightarrow$

$(\mathsf{lock} \wedge \mathsf{offer} \wedge ((\mathsf{lock} \rightarrow \mathsf{send}) \vee \mathsf{abort}))$. By PrePost, we weaken it, obtaining $(\mathsf{send} \vee \mathsf{unblock}) \twoheadrightarrow (\mathsf{send} \vee (\mathsf{lock} \wedge \mathsf{abort}))$. By *Bank* and 4.3, we have $(\mathsf{lock} \wedge \mathsf{abort}) \rightarrow \mathsf{unlock}$, as well as $(\mathsf{lock} \wedge \mathsf{abort}) \rightarrow (\mathsf{abort} \wedge \mathsf{unlock})$. Therefore, by PrePost we have $(\mathsf{send} \vee \mathsf{unlock}) \twoheadrightarrow (\mathsf{send} \vee (\mathsf{abort} \wedge \mathsf{unlock}))$. We conclude by PrePost and Fix.

Example 5.4. (Dining retailers) Around a round table, a group of n cutlery retailers is about to have dinner. In the center of the table, there is a large dish of food. Despite the food being delicious, the retailers cannot start eating right now. To do that, and follow the proper etiquette, each retailer needs to have a complete cutlery set, consisting of n pieces, each of a different kind. Each one of the n retailers owns a distinct set of n piece of cutlery, all of the same kind. The retailers start discussing about trading their cutlery, so that they can finally eat. Since everyone wants to get a fair deal, they want to formalize their committments.

We formalize the scenario as follows. Number the retailers $r_1, \ldots, r_n$ together with the kinds of pieces of cutlery, so that r_i initially owns n pieces of kind number i. Then, write $\mathsf{g}_{i,j}$ for "r_i *gives* a piece (of kind i) to r_j". Since retailers can use their own cutlery, we assume $\mathsf{g}_{i,i}$ to be true. Retailer r_i can start *eating* whenever $e_i = \bigwedge_j \mathsf{g}_{j,i}$.

Suppose that r_1 commits to a simple exchange with r_2: they commit to $\mathsf{g}_{2,1} \twoheadrightarrow \mathsf{g}_{1,2}$ and $\mathsf{g}_{1,2} \twoheadrightarrow \mathsf{g}_{2,1}$, and the exchange takes place since $\mathsf{g}_{2,1} \wedge \mathsf{g}_{1,2}$ can be derived. While this seems a fair deal, it actually exposes r_1 to a risk: if $r_3, \ldots, r_n$ perform a similar exchange with r_2, then we have $\mathsf{g}_{2,i} \wedge \mathsf{g}_{i,2}$ for all i. In particular, $\mathsf{g}_{i,2}$ holds for all i, so r_2 can start eating. This is however not necessarily the case for r_1, since r_3 has not committed to any exchange with r_1.

A wise retailer would then never agree to a simple exchange $\mathsf{g}_{2,1} \twoheadrightarrow \mathsf{g}_{1,2}$. Instead, the retailer r_1 could commit to a safer contract[1]:

$$\mathsf{g}_{1,1} \wedge \mathsf{g}_{2,1} \wedge \cdots \wedge \mathsf{g}_{n,1} \twoheadrightarrow \mathsf{g}_{1,1} \wedge \mathsf{g}_{1,2} \wedge \cdots \wedge \mathsf{g}_{1,n}$$

The idea is simple: r_1 requires each piece of cutlery, that is, r_1 requires to be able to start eating (e_1). When this happens, r_1 agrees to provide each other retailer with a piece of his cutlery. Now, assume ech retailer r_i commits to the analogous contract:

$$c_i = e_i \twoheadrightarrow \bigwedge_j \mathsf{g}_{i,j}$$

We can now verify that $\bigwedge_i c_i \rightarrow \bigwedge_i e_i$, that is, the above contracts actually allow everyone to eat. Assume c_i for all i, and define $p_i = \bigwedge_j \mathsf{g}_{i,j}$. Clearly, $c_i = e_i \twoheadrightarrow p_i$. Note that

$$\bigwedge_{j \neq i} p_j = \bigwedge_{j \neq i} \bigwedge_k \mathsf{g}_{j,k} \quad \rightarrow \quad \bigwedge_j \mathsf{g}_{j,i} = e_i \tag{30}$$

since we can choose $k = i$ and $\mathsf{g}_{i,i}$ is true. Therefore, by PrePost,

$$c_i = e_i \twoheadrightarrow p_i \quad \rightarrow \quad \Big(\bigwedge_{j \neq i} p_j\Big) \twoheadrightarrow p_i$$

By Lemma 4.10, since $\bigwedge_i c_i$, we have $\bigwedge_i p_i$. By (30) we then conclude $\bigwedge_i e_i$.

[1] We include $\mathsf{g}_{1,1} = \top$ to make it more homogeneous.

Example 5.5. In Sect. 3 we listed several requirements for the notion of contractual implication. A small set of these was then picked as our axiomatization: Zero, Fixand PrePost. However, we have not yet checked that this is indeed complete with respect to the other requirements. That is, we have to check that all the requirements are theorems of PCL. We now quickly recall the list of all these requirements and provide a support for each one.

The handshaking properties (1,2) have been established in Lemma 4.9. Property (3) is the axiom Fix. The "greedy handshaking" (4) was established in Lemma 4.10. Property (6) was proved in Lemma 4.3. Property (7) is a direct consequence of Zero. Property (8) is proved in Lemma 4.4. Properties (9,10) are direct consequences of PrePost. Property (11) is a consequence of axioms Zero and PrePost.

6 Proof Theory: Sequent Calculus

In this section we provide an alternative formalization of PCL, through a sequent calculus à la Gentzen. Our sequents have the form $\Gamma \vdash p$, where Γ is a finite set of formulae. Below, we write Γ, p for $\Gamma \cup \{p\}$.

Most of the rules for the IPC fragment have been taken from [27], which features a rule set for IPC without structural rules. This fact turns out to be quite useful when reasoning about the rule system. Indeed, as in [27], we are able to establish cut-elimination by applying a reasonably simple structural induction; we refer to Sect. 7 for the detailed proof. In the IPC fragment of our rule system below, only rules $\neg R$ and $weakR$ diverge from [27]. This change was required to establish the subformula property (Lemma 6.13). Another minor difference from [27] arises from our Γ's being sets rather than multisets; this is however immaterial, because of the absence of structural rules, and the admissibility of contraction in [27].

Definition 6.1. The Gentzen-style rule system for PCL comprises the following rules.

IPC core rules

$$\frac{}{\Gamma, p \vdash p}\,\mathsf{id}$$

$$\frac{\Gamma, p \wedge q, p \vdash r}{\Gamma, p \wedge q \vdash r} \wedge L1 \quad \frac{\Gamma, p \wedge q, q \vdash r}{\Gamma, p \wedge q \vdash r} \wedge L2 \quad \frac{\Gamma \vdash p \;\; \Gamma \vdash q}{\Gamma \vdash p \wedge q} \wedge R$$

$$\frac{\Gamma, p \vee q, p \vdash r \;\; \Gamma, p \vee q, q \vdash r}{\Gamma, p \vee q \vdash r} \vee L \quad \frac{\Gamma \vdash p}{\Gamma \vdash p \vee q} \vee R1 \quad \frac{\Gamma \vdash q}{\Gamma \vdash p \vee q} \vee R2$$

$$\frac{\Gamma, p \to q \vdash p \;\; \Gamma, p \to q, q \vdash r}{\Gamma, p \to q \vdash r} \to L \quad \frac{\Gamma, p \vdash q}{\Gamma \vdash p \to q} \to R$$

$$\frac{\Gamma, \neg p \vdash p}{\Gamma, \neg p \vdash r} \neg L \quad \frac{\Gamma, p \vdash \bot}{\Gamma \vdash \neg p} \neg R$$

$$\frac{}{\Gamma, \bot \vdash p} \bot L \quad \frac{}{\Gamma \vdash \top} \top R \quad \frac{\Gamma \vdash \bot}{\Gamma \vdash p}\,\mathsf{weakR}$$

$$\frac{\Gamma \vdash p \;\; \Gamma, p \vdash q}{\Gamma \vdash q}\,\mathsf{cut}$$

Contractual implication rules

$$\frac{\Gamma \vdash q}{\Gamma \vdash p \twoheadrightarrow q}\,\mathsf{Zero}$$

$$\frac{\Gamma, p \twoheadrightarrow q, r \vdash p \;\; \Gamma, p \twoheadrightarrow q, q \vdash r}{\Gamma, p \twoheadrightarrow q \vdash r}\,\mathsf{Fix}$$

$$\frac{\Gamma, p \twoheadrightarrow q, a \vdash p \;\; \Gamma, p \twoheadrightarrow q, q \vdash b}{\Gamma, p \twoheadrightarrow q \vdash a \twoheadrightarrow b}\,\mathsf{PrePost}$$

Note that IPC rules have left and right rules for each IPC connective. Roughly, each connective has right rules for introduction, and left rules for elimination. Contractual implication instead has a different flavor. Rule Zero is effectively an introduction rule, and has a similar function to the Hilbert axiom Zero. Similarly, rule Fix is an elimination rule; its function is related to the Hilbert axiom Fix. Indeed, these rules could be named $\twoheadrightarrow R$ and $\twoheadrightarrow L$, respectively. The left/right rule dualism however is broken by rule PrePost. Of course, its function is that of the Hilbert axiom PrePost. This rule behaves both as an introduction and an elimination, so is both a right and left rule, in a sense.

In the rest of this section we study the sequent rule system above. First, we prove it equivalent to our Hilbert axioms (Th. 6.4). Then we prove the redundancy of the weakR rule, i.e. that weakR is admissible in the proof system composed of all the other rules. More importantly, we also prove the cut rule redundant, i.e. we prove cut-elimination for our sequent rule system. As a consequence of cut-elimination, we are able to establish the consistency of the logic (Th. 6.11), the subformula property (Lemma 6.13), and the decidability

of $\vdash$(Lemma 6.14). We also prove that the rules Zero, PrePost, Fix are not redundant, as one might expect (Lemma 6.15). Finally we prove that PCL is a conservative extension of IPC (Lemma 6.16).

6.1 Equivalence of the Hilbert and Gentzen Systems

In this section, we establish the equivalence between the two different logical systems we introduced. We denote with $\vdash_H p$ the fact that p is provable from the Hilbert-style axioms of Def. 4.2. Similarly, by $\vdash_G p$ we denote that $\emptyset \vdash p$ is a derivable sequent from the Gentzen-style rules of Def. 6.1. We will then prove $\vdash_H = \vdash_G$.

Lemma 6.2. $\vdash_H p \implies \vdash_G p$

Proof. It is sufficient to show that $\vdash_G$ holds for all the Hilbert-style axioms, and that it is closed under modus ponens. The latter is trivially done by cut and $\to R$. For the former, we check the $\twoheadrightarrow$-related axioms, only, since the others are standard.

- Zero

$$\dfrac{\dfrac{}{\vdash \top}\top R}{\vdash \top \twoheadrightarrow \top}\textsf{Zero}$$

- PrePost

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Delta, p' \vdash p'}id \quad \dfrac{}{\Delta, p', p \vdash p}id}{\Delta, p' \vdash p} \to L \quad \dfrac{\dfrac{}{\Delta, q \vdash q'}id \quad \dfrac{}{\Delta, q, q' \vdash q'}id}{\Delta, q \vdash q'} \to L}{\Delta = p' \to p, p \twoheadrightarrow q, q \to q' \vdash p' \twoheadrightarrow q'}\textsf{PrePost}}{p' \to p, p \twoheadrightarrow q \vdash (q \to q') \to (p' \twoheadrightarrow q')} \to R}{p' \to p \vdash (p \twoheadrightarrow q) \to (q \to q') \to (p' \twoheadrightarrow q')} \to R}{\vdash (p' \to p) \to (p \twoheadrightarrow q) \to (q \to q') \to (p' \twoheadrightarrow q')} \to R$$

- Fix

$$\dfrac{\dfrac{\dfrac{}{p \twoheadrightarrow p, p \vdash p}id \quad \dfrac{}{p \twoheadrightarrow p, p \vdash p}id}{p \twoheadrightarrow p \vdash p}\textsf{Fix}}{\vdash (p \twoheadrightarrow p) \to p} \to R$$

□

Lemma 6.3. $\vdash_G p \implies \vdash_H p$

Proof. It is sufficient to prove the following statement for each rule:

$$\dfrac{\Gamma_0 \vdash_G p_0 \quad \cdots \quad \Gamma_n \vdash_G p_n}{\Gamma \vdash_G p} \implies \vdash_H \bigwedge_i [\bigwedge \Gamma_i \to p_i] \to \bigwedge \Gamma \to p$$

Then, the lemma follows by induction on the derivation of $\vdash_G p$. Most cases are standard, so we check only the $\twoheadrightarrow$-related rules.

- Rule Zero: $(\bigwedge \Gamma \to q) \to \bigwedge \Gamma \to (p \twoheadrightarrow q)$.
  Assume the hypotheses. By modus ponens, we get q, hence $\top \to q$. We have $\top \twoheadrightarrow \top$ by Zero. The formula $p \to \top$ trivially holds. We then apply PrePost to reach $p \twoheadrightarrow q$:

$$(p \to \top) \to (\top \twoheadrightarrow \top) \to (\top \to q) \to (p \twoheadrightarrow q)$$

- Rule PrePost:$[(\bigwedge \Gamma \wedge (p \twoheadrightarrow q) \wedge a) \rightarrow p] \wedge [(\bigwedge \Gamma \wedge (p \twoheadrightarrow q) \wedge b) \rightarrow q] \rightarrow (\bigwedge \Gamma \wedge (p \twoheadrightarrow q)) \rightarrow (a \twoheadrightarrow b)$.
  Assume all the hypotheses. We easily get $a \rightarrow p, p \twoheadrightarrow q, q \rightarrow b$. By PrePost, we get $a \twoheadrightarrow b$.
- Rule Fix: $[(\bigwedge \Gamma \wedge (p \twoheadrightarrow q) \wedge r) \rightarrow p] \wedge [(\bigwedge \Gamma \wedge (p \twoheadrightarrow q) \wedge q) \rightarrow r] \rightarrow (\bigwedge \Gamma \wedge (p \twoheadrightarrow q)) \rightarrow r$.
  Assume all the hypotheses. We get $r \rightarrow p, q \rightarrow r$, hence $q \rightarrow p$. Using $q \rightarrow p$ (deduced), $p \twoheadrightarrow q$ (hypothesis), $q \rightarrow q$ (trivial), we apply PrePost and get $q \twoheadrightarrow q$. By Fix, q, hence r.

□

Theorem 6.4. $\vdash_G = \vdash_H$

Proof. Immediate from lemmas 6.2 and 6.3. □

6.2 Properties of the Gentzen System

A first basic result of our system is that the left-weakening of a sequent, i.e. augmenting the Γ, is strongly admissible. That is, whenever we have a derivation for a sequent, we can produce a derivation for the augmented sequent having the same height.

Lemma 6.5. *If* $\dfrac{D}{\Gamma \vdash p}$ *then* $\dfrac{D'}{\Gamma, \Gamma' \vdash p}$ *where* D' *has the same height of* D.

Proof. It is sufficient to augment each sequent in D with Γ'. It is straightforward to check that no rule is invalidated by this. □

Convention. The above lemma is very frequently used in our proofs. To avoid continuously referring to it, we adopt the following notation: when we have $\dfrac{D}{\Gamma \vdash p}$, we simply write $\dfrac{D+}{\Gamma, \Gamma' \vdash p}$ for the augmented derivation.

The next result is dual to Lemma 6.5. It states that the right-weakening of a sequent, i.e. replacing a $\bot$ on the right of the turnstile $\vdash$ with some other formula , can be performed without using the weakR rule. In other words, rule weakR is redundant in our system. To prove this, we first introduce an auxiliary lemma.

Lemma 6.6. *If* $\dfrac{D}{\Gamma \vdash \bot}$ *where* D *is a* weakR*-free derivation, then we also have* $\Gamma \vdash p$ *with a* weakR*-free derivation, for any* p.

Proof. By induction on the height of the derivation D. The last step of D must be one of id, cut, Fix or a left rule. If a left rule or a cut has been used, the thesis is either trivial ($\neg L, \bot L$) or immediately follows by the induction hypothesis. If id has been used, then $\bot \in \Gamma$, and $\bot L$ suffices. If Fix has been used, we rewrite the derivation in the following way:

$$\dfrac{\dfrac{D0}{\Gamma, a \twoheadrightarrow b, \bot \vdash a} \quad \dfrac{D1}{\Gamma, a \twoheadrightarrow b, b \vdash \bot}\mathsf{Fix}}{\Gamma, a \twoheadrightarrow b \vdash \bot} \implies$$

$$\dfrac{\dfrac{\dfrac{}{\Gamma, a \twoheadrightarrow b, p, a \vdash a}id \quad \dfrac{D1''+}{\Gamma, a \twoheadrightarrow b, p, b \vdash a}}{\Gamma, a \twoheadrightarrow b, p \vdash a}\mathsf{Fix} \quad \dfrac{D1'}{\Gamma, a \twoheadrightarrow b, b \vdash p}}{\Gamma, a \twoheadrightarrow b \vdash p}\mathsf{Fix}$$

where $D1'$ and $D1''$ are obtained from the induction hypothesis on $D1$, and the formulae p, a respectively. □

Note that the transformation above does not introduce new cuts in the derivation. So, once the cut has been eliminated, the same procedure can eliminate weakR, too.

Lemma 6.7. *The* weakR *rule is redundant.*

Proof. It is sufficient to iterate Lemma 6.6. □

Another fundamental result enjoyed by our Gentzen-style rules, is the redundancy of the cut rule. This is a classic cut-elimination result, or *Haupsatz*, which is the basis for many of the results in this section.

Theorem 6.8. (Cut-elimination) *The* cut *rule is redundant.*

Proof. See Sect. 7 for the detailed proof. □

It is possible to remove both the rules cut and weakR from our system without affecting the generated $\vdash$ relation.

Theorem 6.9. *The rule set* $\{\mathsf{cut}, \mathsf{weakR}\}$ *is redundant.*

Proof. This is *not* an immediate corollary of the previous results, since removing a *cut* from a derivation could force us to include weakR in the new derivation, and, vice versa, removing a weakR could force us to introduce a cut. However, by inspecting the proof for Lemma 6.7, we can see that the weakR-elimination procedure does not introduce new *cut*s. So, given an arbitrary derivation D for a sequent, by Th. 6.8 we also have a cut-free derivation D' for the same sequent. Then, we can apply the procedure of Lemma 6.7 to conclude. □

In our logic, as for IPC, every negation-free theory Γ is consistent.

Lemma 6.10. *Let* Γ *be free from* $\neg, \bot$. *Then* $\Gamma \nvdash \bot$.

Proof. By contradiction, assume $\dfrac{D}{\Gamma \vdash \bot}$ to be a cut-free derivation. We proceed by induction on the derivation D, and then by case analysis. The last step of D can not be $id, \neg L, \bot L$, otherwise Γ is not free from $\neg, \bot$. If the last step is another left rule, we have $\Gamma' \vdash \bot$ as a premise for some Γ' free from $\neg, \bot$ so the inductive hypothesis suffices. The same applies to Fix. No right rule can introduce $\bot$, so the last step is not a right rule. The same applies to Zero, PrePost. No other rule exists. □

Theorem 6.11. (Consistency) *The logic is consistent, i.e.* $\emptyset \nvdash \bot$.

Proof. Direct consequence of Lemma 6.10. □

Cut-free derivations enjoy the *subformula* property, stating that all the formulae occurring such a derivation for a sequent appear as subformulae in the sequent as well. Equivalently, it states that a cut-free derivation of a sequent can only involve subformulae of that sequent. As a single exception, the derivation might mention $\bot$ while the sequent does not, because of the weakR rule.

Definition 6.12. The subformulae $sub(p)$ of a formula p are inductively defined as follows

$$
\begin{aligned}
&sub(\mathsf{p}) = \{\mathsf{p}\}\\
&sub(\bot) = \{\bot\}\\
&sub(\top) = \{\top\}\\
&sub(\neg p) = \{\neg p\} \cup sub(p)\\
&sub(p \vee q) = \{p \vee q\} \cup sub(p) \cup sub(q)\\
&sub(p \wedge q) = \{p \wedge q\} \cup sub(p) \cup sub(q)\\
&sub(p \rightarrow q) = \{p \rightarrow q\} \cup sub(p) \cup sub(q)\\
&sub(p \twoheadrightarrow q) = \{p \twoheadrightarrow q\} \cup sub(p) \cup sub(q)
\end{aligned}
$$

The subformulae of a set of formulae Γ is $sub(\Gamma) = \bigcup_{p \in \Gamma} sub(p)$.

Lemma 6.13. (Subformula Property) *If* $\dfrac{D}{\Gamma \vdash p}$ *and* D *is cut-free, then the formulae occurring in* D *belong to* $sub(\Gamma, p, \bot)$.

Proof. By a simple induction on D. The property is preserved by every rule. □

Note that a $\{\mathsf{cut}, \mathsf{weakR}\}$-free derivation would instead have a more tight $sub(\Gamma, p)$ bound.

We can now establish decidability for PCL.

Lemma 6.14. (Decidability) $\Gamma \vdash p$ *is decidable.*

Proof. We have $\Gamma \vdash p$ iff it can be derived without cuts. We can decide the latter by searching for a shortest derivation bottom-up, exploring the whole proof space non-deterministically. By the subformula property, cut-free proofs can only contain sequents having formulae in $sub(\Gamma, p, \bot)$. This is a finite set of sequents: let k be its cardinality. The depth of the search in the proof space can be limited to k: if there is a taller derivation, it has a sequent occurring twice in some path, so the proof can be made shorter. This ensures the termination of the algorithm. □

We have implemented the above naïve decision procedure for PCL, developing a prototype tool, which we used for experimenting with our logic. Since the proof space is huge, the tool was very helpful in establishing the negative results for our logic, i.e. $\nvdash p$ from some given p. We give more information about it in Sect. A.2.

We can now prove the non redundancy of the $\twoheadrightarrow$-related rules.

Lemma 6.15. *None of the* Zero, PrePost, Fix *Gentzen rules is redundant.*

Proof. First, we carefully examine the proof of the cut-elimination theorem, and check that the same proof actually establish cut-elimination event in the rule system without any of the rules Zero, PrePost, Fix. This is because the cut-elimination procedure never introduces a new application of, say, a Fix rule unless Fix was already present in the original derivation. Similarly, we can restate the subformula property in these restricted rule systems, as well as decidability. We therefore have decision procedures for both the full rule system and each restriction of it, so we can check that indeed some formula is provable in the full system, but not in the restriction. As it might be expected, it turns out that to prove the Hilbert axioms Zero, PrePost, Fix in the Gentzen system, the related rule is necessary. This was checked by a simple modification of our tool, described in Sect. A.2. □

As a nice result of the subformula property, we get that PCL is a conservative extension of IPC.

Lemma 6.16. PCL *is a conservative extension of IPC, that is* $\vdash_{IPC} p \iff \vdash p$ *for all IPC formulae* p.

Proof. The $\Rightarrow$ part is Lemma 8.4. For the $\Leftarrow$ part, if $\vdash p$, by Lemma 6.2 we have $\vdash_G p$. By cut-elimination and the subformula property, we have $\dfrac{D}{\vdash p}$ where D make no use of rules Zero, Fix, and PrePost. Since all the other rules are included in the Gentzen system of IPC, we can state $\vdash_{IPC} p$. □

7 Cut-elimination

In this section we prove cut elimination for the Gentzen system of Def. 6.1. In order to do this, we borrow a fairly simple structural induction technique from [27].[2] The whole technique can be described as a recursive algorithm, transforming a generic derivation into a cut-free one. The algorithm is made of two recursive routines: a core routine (CUT-REDUCE) and a driver (CUT-ELIM). The core routine deals with the special case of a derivation ending with a *cut* between two *cut-free* subderivations. We name derivations of this special form *reducible* derivations. When provided with a reducible derivation, CUT-REDUCE produces a cut-free derivation for the same sequent. Exploiting CUT-REDUCE, the driver CUT-ELIM handles the general case. The actual procedure is shown in Alg. 1.

[2] Here, "structural" means that it proceeds inductively on the structure of a derivation in the Gentzen system.

Algorithm 1 Main driver for cut-elimination.

$$\textsc{cut-elim}\left(\dfrac{\dfrac{D_0}{\Gamma_0 \vdash p_0} \quad \dfrac{D_1}{\Gamma_1 \vdash p_1}}{\Gamma \vdash p}\mathit{cut}\right) = \textsc{cut-reduce}\left(\dfrac{\dfrac{D'_0}{\Gamma_0 \vdash p_0} \quad \dfrac{D'_1}{\Gamma_1 \vdash p_1}}{\Gamma \vdash p}\mathit{cut}\right)$$

$$\textsc{cut-elim}\left(\dfrac{\cdots \dfrac{D_i}{\Gamma_i \vdash p_i} \cdots}{\Gamma \vdash p} r\right) = \dfrac{\cdots \dfrac{D'_i}{\Gamma_i \vdash p_i} \cdots}{\Gamma \vdash p} r \qquad (r \neq \mathit{cut})$$

where $\dfrac{D'_i}{\Gamma_i \vdash p_i} = \textsc{cut-elim}\left(\dfrac{D_i}{\Gamma_i \vdash p_i}\right)$

The CUT-ELIM driver takes as input a derivation D. First, it applies itself recursively on the derivations of the premises D_i so converting them to the cut-free ones D'_i. Then, if the last rule r used in D is not a *cut*, we can apply the same rule to D'_i to construct a cut-free derivation. Otherwise, r is a *cut*, and we are in the special case handled by CUT-REDUCE, so we just invoke it. The CUT-ELIM driver always terminates since each recursive call is made on a subderivation.

In the rest of this section, we define the core routine CUT-REDUCE. This routine makes use of a sophisticated recursion scheme. When invoked as

$$\textsc{cut-reduce}\left(\dfrac{\dfrac{D_0}{\Gamma \vdash p} \quad \dfrac{D_1}{\Gamma, p \vdash q}}{\Gamma \vdash q}\mathit{cut}\right) \quad \text{with cut-free } D_0, D_1$$

the routine makes several recursive calls, all of them for reducible derivations. Assume a recursive call is made on a cut having p' as the cut formula, and D'_0, D'_1 as the subderivations. Then, one of the following conditions holds:

Definition 7.1. Constraints on the recursive calls to CUT-REDUCE:[3]

1. p' is a proper subformula of p, or
2. $p' = p$ and $h(D'_0) < h(D_0)$, or
3. $p' = p$, $h(D'_0) \leq h(D_0)$, and $h(D'_1) < h(D_1)$.

We can state the conditions above as follows: the triple $(p', h(D'_0), h(D'_1))$ is lexicographically smaller than $(p, h(D_0), h(D_1))$. It is easy to see that this ordering for triples is a well-founded ordering relation, and therefore the recursion must eventually terminate.

We now proceed to define the CUT-REDUCE routine. This is done by examining the last rules used in D_0 and D_1, and covering all the possible cases. To simplify the presentation, we adopt a compact notation, writing $\Longrightarrow$ for our reduction, as seen in Alg. 2. The rest of this section will precisely define the $\Longrightarrow$ relation.

[3] We let conditions 2 and 3 to overlap, since we use $\leq$ instead of $=$. This is done to follow our actual reduction procedure more closely, as we will see.

Algorithm 2 Core routine for cut-elimination.

$$\textsc{cut-reduce}\left(\dfrac{\dfrac{D_0}{\Gamma \vdash p} \quad \dfrac{D_1}{\Gamma, p \vdash q}}{\Gamma \vdash q}\mathit{cut}\right) = \bar{D}$$

where $\dfrac{\dfrac{D_0}{\Gamma \vdash p} \quad \dfrac{D_1}{\Gamma, p \vdash q}}{\Gamma \vdash q}\mathit{cut} \implies D'$

and $\bar{D}$ is obtained by recursively applying CUT-REDUCE to all the *cut*s in D'.

When we write $D \Longrightarrow D'$, D is a reducible derivation, and D' is the result of the reduction. When we use *cut*s in D', they are to be interpreted as recursive calls to the CUT-REDUCE routine. Of course, we shall take care these cuts agree with the well-founded ordering discussed above. We document this in the derivation D', by writing cut_p when we are in case 1 of the enumeration above, cut_0 when we are in case 2, or cut_1 when we are in case 3.

7.1 Summary of Cases

To classify all possible cases, we first introduce some terminology. Each Gentzen rule is related to some logical connective. A rule for a generic connective $\odot$ has as its principal formulae the formulae occurring in that rule that involve $\odot$. Rules $\mathit{id}, \mathit{weakR}$ and *cut* have no principal formulae. Both "left" and "right" rules in the IPC fragment have exactly one principal formula. Rules Zero and Fix have one principal formula as well. When a principal formula occurs in the left (resp. right) hand side of the turnstile $\vdash$we call it a left (resp. right) principal formula. Rule PrePost has two principal formulae, named left principal and right principal formulae.

In order to reduce:

$$\dfrac{\dfrac{D_0}{\Gamma \vdash a} \quad \dfrac{D_1}{\Gamma, a \vdash b}}{\Gamma \vdash b}\mathit{cut}$$

we proceed by case analysis on the last rule used in D_0, D_1. The derivation D_0can end with a Zero rule, a PrePost rule, a Fix rule, or an IPC rule. Similarly for D_1. These 4^2cases are further split, according to whether

- the cut formula is the left principal formula of D_0 and the right principal formula of D_1 (the *essential* case),
- is *not* the right principal formula of D_0 (the *left commutation* case), or
- is *not* the left principal formula of D_1 (the *right commutation* case).

This classification is rather standard in cut-elimination results, and is used in [27] as well. Note that the two commutation cases can overlap when the cut formula is not right/left principal in both D_0, D_1, respectively. In Table 1, we cover all the possible cases, and group them whenever the handling is similar.

We now provide a reading key for Table 1. The first row describes the case where D_0 ends with Zero. In this case, the cut formula is the right principal formula of D_0, and so there is no left commutation case, denoted by $\nexists$(l). The first

$D_0 \backslash\ D_1$	Zero	PrePost	Fix	IPC
Zero	$\nexists$ (e)	Zero/PrePost (e)	Zero/Fix (e)	$\nexists$ (e)
$\nexists$ (l)	*/Zero (r)	*/PrePost (r)	*/Fix (r)	standard (r)
PrePost	$\nexists$ (e)	PrePost/PrePost (e)	PrePost/Fix (e)	$\nexists$ (e)
$\nexists$ (l)	*/Zero (r)	*/PrePost (r)	*/Fix (r)	standard (r)
Fix	[Fix/* (l)]	Fix/* (l)	Fix/* (l)	Fix/* (l)
$\nexists$ (e)	*/Zero (r)	[*/PrePost (r)]	[*/Fix (r)]	[standard (r)]
	$\nexists$ (e)	$\nexists$ (e)	$\nexists$ (e)	
IPC	[standard (l)]	standard (l)	standard (l)	standard (e,l,r)
	*/Zero (r)	[*/PrePost (r)]	[*/Fix (r)]	

(e) essential, (l) left commutation, (r) right commutation, [subsumed]

Table 1: Summary of cases for cut-elimination

cell is for the case where D_1 ends with Zero as well: this is a right commutation case which is handled in the same way as PrePost/Zero, Fix/Zero, etc. so we will have a single case $*$/Zero case for the whole column. The Zero/PrePost case can be an essential case or a right commutation depending on whether the cut formula is the left principal formula of PrePost, so we split in two subcases. The right commutation case $*$/PrePost is also reused in other points in the same column. The same applies to Zero/Fix. The Zero/IPC case can not be an essential case, since $\twoheadrightarrow$ never occurs in an IPC rule; so this is a right commutation.

The second row describes the case where D_0 ends with PrePost, and is similar to the first row.

The third row describes the case where D_0 ends with Fix. Since Fix has no right principal formula, there is no essential case in this row, denoted by $\nexists$(e). The case Fix/Zero is both a left and right commutation case, which we arbitrarily chose to handle as a right commutation case. The remaining cases *might* be right commutations, but are surely left commutations Fix/$*$, so we handle them in that way.

The fourth row describes the case where D_0 ends with an IPC rule. The case IPC/Zero *might* be a left commutation (depending on the actual IPC rule), but is surely a right commutation as well, so we handle it in that way. The case IPC/PrePost can not be an essential case, since $\twoheadrightarrow$ occurs in no IPC rule; it *might* be a right commutation, but is surely a left commutation (no IPC rules involves $\twoheadrightarrow$), so we handle it in that way. The case IPC/Fix is similar. The case IPC/IPC can be either essential, a left commutation, or a right commutation.

We invite the reader to check that Table 1 indeed enumerates all the possible cases, which can therefore be grouped as follows:

- essential: Zero/PrePost (e), Zero/Fix (e), PrePost/PrePost (e), PrePost/Fix (e), standard (e)
- left commutations: Fix/$*$ (l), standard (l)
- right commutations: $*$/Zero (r), $*$/PrePost (r), $*$/Fix (r), standard (r)

Most cases of the "standard" group are well-known cases for IPC, and are covered in [27], Appendix 1. This includes, for instance, the essential case $\wedge R/\wedge L1$, the left commutation $\wedge L1/*$, and the right commutation $*/\wedge R$. Handling these cases here would essentially amount to copying the whole Appendix 1 of [27]

here, and perform some minor notation change, only. Since this would provide no actual contribution, we will refer to [27] when these cases arise, and omit them. For completeness, we document in Sect. A.1 how to rephrase [27] in our notation. Finally, recall that in Def. 6.1 we diverge from [27] in a few IPC rules, namely $\neg R$ and $weakR$. Of course, these rules are involved in standard cases which are not covered in [27], so we shall provide a reduction for these cases. The case $\neg R/*$ can not be a left commutation, but it can be essential ($\neg R/\neg L$); the case $*/\neg R$ is instead a right commutation. The case $*/weakR$ is a right commutation, while $weakR/*$ is a left commutation.

We now proceed by handling all the cases mentioned above. We sometimes write $\Gamma(p)$ instead of Γ to stress that $p \in \Gamma$.

7.2 The Essential Cases

In these cases the cut formula is (right/left) principal in both the premises of the cut.

- Case Zero/PrePost

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash q}D0}{\Gamma \vdash p \twoheadrightarrow q}\mathsf{Zero} \quad \dfrac{\dfrac{}{\Gamma, p \twoheadrightarrow q, a \vdash p}D1 \quad \dfrac{}{\Gamma, p \twoheadrightarrow q, q \vdash b}D2}{\Gamma, p \twoheadrightarrow q \vdash a \twoheadrightarrow b}\mathsf{PrePost}}{\Gamma \vdash a \twoheadrightarrow b}cut \quad \Longrightarrow$$

$$\dfrac{\dfrac{}{\Gamma \vdash q}D0 \quad \dfrac{\dfrac{\dfrac{}{\Gamma, q \vdash q}id}{\Gamma, q \vdash p \twoheadrightarrow q}\mathsf{Zero} \quad \dfrac{}{\Gamma, q, p \twoheadrightarrow q \vdash b}D2}{\Gamma, q \vdash b}cut_1}{\dfrac{\Gamma \vdash b}{\Gamma \vdash a \twoheadrightarrow b}\mathsf{Zero}}cut_p$$

- Case Zero/Fix

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash q}D0}{\Gamma \vdash p \twoheadrightarrow q}\mathsf{Zero} \quad \dfrac{\dfrac{}{\Gamma, p \twoheadrightarrow q, a \vdash p}D1 \quad \dfrac{}{\Gamma, p \twoheadrightarrow q, q \vdash a}D2}{\Gamma, p \twoheadrightarrow q \vdash a}\mathsf{Fix}}{\Gamma \vdash a}cut \quad \Longrightarrow$$

$$\dfrac{\dfrac{}{\Gamma \vdash q}D0 \quad \dfrac{\dfrac{\dfrac{}{\Gamma, q \vdash q}id}{\Gamma, q \vdash p \twoheadrightarrow q}\mathsf{Zero} \quad \dfrac{}{\Gamma, q, p \twoheadrightarrow q \vdash a}D2}{\Gamma, q \vdash a}cut_1}{\Gamma \vdash a}cut_p$$

- Case PrePost/PrePost. We can assume $(p \twoheadrightarrow q) \in \Gamma$.

$$\dfrac{\dfrac{\dfrac{D0}{\Gamma, a \vdash p} \quad \dfrac{D1}{\Gamma, q \vdash b}\,\mathsf{PrePost}}{\Gamma \vdash a \twoheadrightarrow b} \quad \dfrac{\dfrac{D2}{\Gamma, a \twoheadrightarrow b, x \vdash a} \quad \dfrac{D3}{\Gamma, a \twoheadrightarrow b, b \vdash y}\,\mathsf{PrePost}}{\Gamma, a \twoheadrightarrow b \vdash x \twoheadrightarrow y}}{\Gamma(p \twoheadrightarrow q) \vdash x \twoheadrightarrow y}\,cut \implies$$

$$\dfrac{\dfrac{\hat{D}_0}{\Gamma, x \vdash p} \quad \dfrac{\hat{D}_1}{\Gamma, q \vdash y}\,\mathsf{PrePost}}{\Gamma(p \twoheadrightarrow q) \vdash r}$$

$$\hat{D}_0 = \dfrac{\dfrac{\dfrac{\dfrac{D0+}{\Gamma, x, a \vdash p} \quad \dfrac{D1+}{\Gamma, x, q \vdash b}\,\mathsf{PrePost}}{\Gamma, x \vdash a \twoheadrightarrow b} \quad \dfrac{D2}{\Gamma, x, a \twoheadrightarrow b \vdash a}\,cut_1}{\Gamma, x \vdash a} \quad \dfrac{D0+}{\Gamma, x, a \vdash p}\,cut_p}{\Gamma, x \vdash p}$$

$$\hat{D}_1 = \dfrac{\dfrac{D1}{\Gamma, q \vdash b} \quad \dfrac{\dfrac{\dfrac{D0+}{\Gamma, q, b, a \vdash p} \quad \dfrac{D1+}{\Gamma, q, b, q \vdash b}\,\mathsf{PrePost}}{\Gamma, q, b \vdash a \twoheadrightarrow b} \quad \dfrac{D3+}{\Gamma, q, b, a \twoheadrightarrow b \vdash y}\,cut_1}{\Gamma, q, b \vdash y}}{\Gamma, q \vdash y}\,cut_p$$

- Case PrePost/Fix. We can assume $(p \twoheadrightarrow q) \in \Gamma$.

$$\dfrac{\dfrac{\dfrac{D0}{\Gamma, a \vdash p} \quad \dfrac{D1}{\Gamma, q \vdash b}\,\mathsf{PrePost}}{\Gamma \vdash a \twoheadrightarrow b} \quad \dfrac{\dfrac{D2}{\Gamma, a \twoheadrightarrow b, r \vdash a} \quad \dfrac{D3}{\Gamma, a \twoheadrightarrow b, b \vdash r}\,\mathsf{Fix}}{\Gamma, a \twoheadrightarrow b \vdash r}}{\Gamma(p \twoheadrightarrow q) \vdash r}\,cut \implies$$

$$\dfrac{\dfrac{\hat{D}_0}{\Gamma, r \vdash p} \quad \dfrac{\hat{D}_1}{\Gamma, q \vdash r}\,\mathsf{Fix}}{\Gamma \vdash r}$$

$$\hat{D}_0 = \dfrac{\dfrac{\dfrac{\dfrac{D0+}{\Gamma, r, a \vdash p} \quad \dfrac{D1+}{\Gamma, r, q \vdash b}\,\mathsf{PrePost}}{\Gamma, r \vdash a \twoheadrightarrow b} \quad \dfrac{D2}{\Gamma, r, a \twoheadrightarrow b \vdash a}\,cut_1}{\Gamma, r \vdash a} \quad \dfrac{D0+}{\Gamma, r, a \vdash p}\,cut_p}{\Gamma, r \vdash p}$$

$$\hat{D}_1 = \dfrac{\dfrac{D1}{\Gamma, q \vdash b} \quad \dfrac{\dfrac{\dfrac{D0+}{\Gamma, q, b, a \vdash p} \quad \dfrac{D1+}{\Gamma, q, b, q \vdash b}\,\mathsf{PrePost}}{\Gamma, q, b \vdash a \twoheadrightarrow b} \quad \dfrac{D3+}{\Gamma, q, b, a \twoheadrightarrow b \vdash r}\,cut_1}{\Gamma, q, b \vdash r}}{\Gamma, q \vdash r}\,cut_p$$

- standard. As anticipated, we refer to [27] here, but for the case $\neg R/\neg L$, shown below.

- Case $\neg R/\neg L$

$$\dfrac{\dfrac{\dfrac{D0}{\Gamma, p \vdash \bot}}{\Gamma \vdash \neg p}\neg R \quad \dfrac{\dfrac{D1}{\Gamma, \neg p \vdash p}}{\Gamma, \neg p \vdash q}\neg L}{\Gamma \vdash q}\mathit{cut} \quad \Longrightarrow$$

$$\dfrac{\dfrac{\dfrac{\dfrac{D0}{\Gamma, p \vdash \bot}}{\Gamma \vdash \neg p}\neg R \quad \dfrac{D1}{\Gamma, \neg p \vdash p}}{\Gamma \vdash p}\mathit{cut}_1 \quad \dfrac{D0}{\Gamma, p \vdash \bot}}{\Gamma \vdash \bot}\mathit{cut}_p}{\Gamma \vdash q}\mathit{weakR}$$

7.3 The Left Commutation Cases

In these cases the cut formula is not a right principal formula in the left premise of the cut.

- Case $\mathsf{Fix}/*$.We can assume $(p \twoheadrightarrow q) \in \Gamma$.

$$\dfrac{\dfrac{\dfrac{D0}{\Gamma, a \vdash p} \quad \dfrac{D1}{\Gamma, q \vdash a}}{\Gamma \vdash a}\mathsf{Fix} \quad \dfrac{D2}{\Gamma, a \vdash b}*}{\Gamma(p \twoheadrightarrow q) \vdash b}\mathit{cut} \quad \Longrightarrow$$

$$\dfrac{\dfrac{\dfrac{}{\Gamma, b, p \vdash p}\mathit{id} \quad \dfrac{\dfrac{D1+}{\Gamma, b, q \vdash a} \quad \dfrac{D0+}{\Gamma, b, q, a \vdash p}}{\Gamma, b, q \vdash p}\mathit{cut}_0}{\Gamma, b \vdash p}\mathsf{Fix} \quad \dfrac{\dfrac{D1}{\Gamma, q \vdash a} \quad \dfrac{D2+}{\Gamma, q, a \vdash b}}{\Gamma, q \vdash b}\mathit{cut}_0}{\Gamma \vdash b}\mathsf{Fix}$$

- standard. As anticipated, we refer to [27] here, but for the case $\mathit{weakR}/*$, shown below.

- Case $\mathit{weakR}/*$

$$\dfrac{\dfrac{\dfrac{D0}{\Gamma \vdash \bot}}{\Gamma \vdash p}\mathit{weakR} \quad \dfrac{D1}{\Gamma, p \vdash q}*}{\Gamma \vdash q}\mathit{cut} \quad \Longrightarrow \quad \dfrac{\dfrac{D0}{\Gamma \vdash \bot}}{\Gamma \vdash q}\mathit{weakR}$$

7.4 The Right Commutation Cases

In these cases the cut formula is not a left principal formula in the right premise of the cut.

- Case $*/\mathsf{Zero}$

$$\dfrac{\dfrac{D0}{\Gamma \vdash a} * \quad \dfrac{\dfrac{D1}{\Gamma, a \vdash q}}{\Gamma, a \vdash p \twoheadrightarrow q}\mathsf{Zero}}{\Gamma \vdash p \twoheadrightarrow q}\mathit{cut} \quad \Longrightarrow$$

$$\dfrac{\dfrac{\dfrac{D0}{\Gamma \vdash a} \quad \dfrac{D1}{\Gamma, a \vdash q}}{\Gamma \vdash q}\mathit{cut}_1}{\Gamma \vdash p \twoheadrightarrow q}\mathsf{Zero}$$

- Case $*/\mathsf{PrePost}$. We can assume $(p \twoheadrightarrow q) \in \Gamma$.

$$\dfrac{\dfrac{D0}{\Gamma \vdash a} * \dfrac{\dfrac{D1}{\Gamma, a, x \vdash p} \quad \dfrac{D2}{\Gamma, a, q \vdash y}}{\Gamma, a \vdash x \twoheadrightarrow y}\mathsf{PrePost}}{\Gamma(p \twoheadrightarrow q) \vdash x \twoheadrightarrow y}\mathit{cut} \implies \dfrac{\dfrac{\dfrac{D0+}{\Gamma, x \vdash a} \quad \dfrac{D1}{\Gamma, x, a \vdash p}}{\Gamma, x \vdash p}\mathit{cut}_1 \quad \dfrac{\dfrac{D0+}{\Gamma, q \vdash a} \quad \dfrac{D2}{\Gamma, q, a \vdash y}}{\Gamma, q \vdash y}\mathit{cut}_1}{\Gamma \vdash x \twoheadrightarrow y}\mathsf{PrePost}$$

- Case $*/\mathsf{Fix}$. We can assume $(p \twoheadrightarrow q) \in \Gamma$.

$$\dfrac{\dfrac{D0}{\Gamma \vdash a} * \dfrac{\dfrac{D1}{\Gamma, a, r \vdash p} \quad \dfrac{D2}{\Gamma, a, q \vdash r}}{\Gamma, a \vdash r}\mathsf{Fix}}{\Gamma(p \twoheadrightarrow q) \vdash r}\mathit{cut} \implies \dfrac{\dfrac{\dfrac{D0+}{\Gamma, r \vdash a} \quad \dfrac{D1}{\Gamma, r, a \vdash p}}{\Gamma, r \vdash p}\mathit{cut}_1 \quad \dfrac{\dfrac{D0+}{\Gamma, q \vdash a} \quad \dfrac{D2}{\Gamma, q, a \vdash r}}{\Gamma, q \vdash r}\mathit{cut}_1}{\Gamma \vdash r}\mathsf{Fix}$$

- standard. As anticipated, we refer to [27] here, but for the cases $*/\mathit{weakR}$ and $*/\neg R$, shown below.

- Case $*/\mathit{weakR}$

$$\dfrac{\dfrac{D0}{\Gamma \vdash p} * \dfrac{\dfrac{D1}{\Gamma, p \vdash \bot}\mathit{weakR}}{\Gamma, p \vdash q}}{\Gamma \vdash q}\mathit{cut} \implies \dfrac{\dfrac{D0}{\Gamma \vdash p} \quad \dfrac{D1}{\Gamma, p \vdash \bot}}{\Gamma \vdash \bot}\mathit{cut}_1\mathit{weakR}$$

- Case $*/\neg R$

$$\dfrac{\dfrac{D0}{\Gamma \vdash p} \quad \dfrac{\dfrac{D1}{\Gamma, p, q \vdash \bot}\neg R}{\Gamma, p \vdash \neg q}}{\Gamma \vdash \neg q}\mathit{cut} \implies \dfrac{\dfrac{D0+}{\Gamma, q \vdash p} \quad \dfrac{D1}{\Gamma, q, p \vdash \bot}}{\Gamma, q \vdash \bot}\mathit{cut}_1\neg R$$

8 Relations with Other Logics

In this section we explore the relationships between PCL and other logics. We are interested in possible mappings, to see if there is some way to encode PCL in some other pre-existing logic. For instance, since PCL is a direct extension of IPC, one might wonder whether the newly introduced connective for contractual implication $\twoheadrightarrow$ can be expressed using IPC connectives. We answer negatively to this question in Sect. 8.1. In Sect. 8.2 we explore some mappings to the modal logic S4, by extending a well-known mapping from IPC to S4.

In this section, when we need to prove tautologies of IPC or S4, we will sometimes resort to an automatic theorem prover. To this purpose, we use the Logics WorkBench (LWB) [22]. The interested reader will find in Sect. A.3 how to obtain the actual proofs generated by LWB.

8.1 Mappings to IPC

As shown by Lemma 4.8, the principle of excluded middle trivializes contractual implication. This supports for the use of IPC, rather then e.g. CPC and S4, as the basis for our logic. We study here the mappings that act homomorphically with respect to each IPC connective.

Definition 8.1. A homomorphic mapping (from PCL to IPC) is a function $m(\bullet)$ such that:

$$m(\mathsf{p}) = \mathsf{p} \qquad (\mathsf{p}\text{ prime})$$
$$m(p \wedge q) = m(p) \wedge m(q)$$
$$m(p \vee q) = m(p) \vee m(q)$$
$$m(p \rightarrow q) = m(p) \rightarrow m(q)$$
$$m(\neg p) = \neg m(p)$$
$$m(\top) = \top$$
$$m(\bot) = \bot$$

A homomorphic mapping $m(\bullet)$ is *complete* if and only if $\vdash p$ implies $\vdash_{IPC} m(p)$, and it is *sound* if and only if $\vdash_{IPC} m(p)$ implies $\vdash p$. Note that $m(p \twoheadrightarrow q)$ is not constrained by the above definition.

We first state some basic properties of homomorphic mappings.

Lemma 8.2. *Let $m(\bullet)$ be a homomorphic mapping. Then:*

$$m(p \leftrightarrow q) = m(p) \leftrightarrow m(q)$$

Proof. Trivial expansion of the syntactic sugar for $\leftrightarrow$. □

Lemma 8.3. *For all homomorphic mappings $m(\bullet)$ and for all IPC formulae p, we have $m(p) = p$.*

Proof. Trivial structural induction. □

The identity is a sound and complete *partial* mapping for IPC formulae.

Lemma 8.4. $\vdash_{IPC} p \implies \vdash p$.

Proof. Trivial, since the Hilbert axioms of PCL include those of IPC. □

We anticipate here a result which will be formally proved in Sect. 6, namely the partial completeness of the identity mapping. Note that this actually makes PCL a conservative extension of IPC.

Lemma 8.5. *For all IPC formulae p, we have $\vdash p \implies \vdash_{IPC} p$.*

Proof. See Lemma 6.16. □

We are aware of several complete but unsound mappings from PCL to IPC. Among these, two are peculiar, in that they provide the strongest and weakest interpretations of the connective$\twoheadrightarrow$ in IPC. The formal justification for this terminology will be clear after Lemma 8.11.

Definition 8.6. The "strongest" interpretation of $\twoheadrightarrow$ in IPC, called $s(\bullet)$, is defined as the homomorphic mapping such that:

$$s(p \twoheadrightarrow q) = s(q)$$

We now prove the completeness of s, as well as its unsoundness.

Lemma 8.7. *For all* PCL *formulae p, we have that $\vdash p \implies \vdash_{IPC} s(p)$. The converse is false, in general.*

Proof. Easy induction on the derivation of $\vdash p$.

- If $\vdash p$ was derived through an IPC axiom, then we use the same axiom to derive $\vdash_{IPC} s(p)$, since s is an homomorphism.
- If $\vdash p$ was derived through a $\twoheadrightarrow$ axiom, we have the following subcases.
  - Zero: $p = \top \twoheadrightarrow \top$. Trivially, $\vdash_{IPC} s(p) = \top$.
  - Fix: $p = (r \twoheadrightarrow r) \to r$. Trivially, $\vdash_{IPC} s(p) = s(r) \to s(r)$.
  - PrePost: $p = (r' \to r) \to (r \twoheadrightarrow q) \to (q \to q') \to (r' \twoheadrightarrow q')$. Then,

    $$\vdash_{IPC} s(p) = (s(r') \to s(r)) \to s(q) \to (s(q) \to s(q')) \to s(q')$$

    is easy. Indeed, if we generalize the above by replacing $s(r), s(r'), s(q), s(q')$ with distinct prime formulae, the formula still holds, as it can be trivially verified either by hand or through an IPC theorem prover. We used LWB for this: see Sect. A.3 for more details.
- If $\vdash p$ was derived through a cut $\dfrac{p \quad p \to q}{q}$, then by inductive hypothesis we have $\vdash_{IPC} s(p)$ and $\vdash_{IPC} s(p \to q)$, the latter being $\vdash_{IPC} s(p) \to s(q)$. We can then conclude by the cut $\dfrac{s(p) \quad s(p) \to s(q)}{s(q)}$.

The converse does not hold in general, e.g. for $p = (r \twoheadrightarrow q) \to q$. □

Below, we introduce the "weakest" interpretation w for contractual implication. This is somehow dual with respect to s, as we will see in Lemma 8.10.

Definition 8.8. The "weakest" interpretation of $\twoheadrightarrow$ in IPC, called $w(\bullet)$, is defined as the homomorphic mapping such that:

$$w(p \twoheadrightarrow q) = (w(q) \to w(p)) \to w(q)$$

We now prove the completeness of w, as well as its unsoundness.

Lemma 8.9. *For all* PCL *formulae p, we have that $\vdash p \implies \vdash_{IPC} w(p)$. The converse is false, in general.*

Proof. Easy induction on the derivation of $\vdash p$.

- If $\vdash p$ was derived through an IPC axiom, then we use the same axiom to derive $\vdash_{IPC} w(p)$, since w is an homomorphism.
- If $\vdash p$ was derived through a $\twoheadrightarrow$ axiom, we have the following subcases.

- Zero: $p = \top \twoheadrightarrow \top$. Trivially, $\vdash_{IPC} w(p) = (\top \to \top) \to \top$.
- Fix: $p = (r \twoheadrightarrow r) \to r$. Then, $\vdash_{IPC} w(p) = ((w(r) \to w(r)) \to w(r)) \to w(r)$ is simple to prove.
- PrePost: $p = (r' \to r) \to (r \twoheadrightarrow q) \to (q \to q') \to (r' \twoheadrightarrow q')$. Then,

$$\vdash_{IPC} w(p) = (w(r') \to w(r)) \to ((w(q) \to w(r)) \to w(q)) \to A$$
$$\text{where } A = (w(q) \to w(q')) \to ((w(q') \to w(r')) \to w(q'))$$

  can be proved in IPC. Indeed, if we generalize the above by replacing $w(r), w(r'), w(q), w(q')$ with distinct prime formulae, the formula still holds, as it can be trivially verified by an IPC theorem prover. We used LWB for this: see Sect. A.3 for more details.

- If $\vdash p$ was derived through a cut $\dfrac{p \quad p \to q}{q}$, then by inductive hypothesis we have $\vdash_{IPC} w(p)$ and $\vdash_{IPC} w(p \to q)$, the latter being $\vdash_{IPC} w(p) \to w(q)$. We can then conclude by the cut $\dfrac{w(p) \quad w(p) \to w(q)}{w(q)}$.

The converse does not hold in general, e.g. for $p = ((q \to r) \to q) \to (r \twoheadrightarrow q)$. □

Below, we relate the mappings s and w, by examining their behaviour on $p \twoheadrightarrow q$, where p, q are IPC formulae. In Lemma 4.7 we proved that these mappings give sufficient and necessary conditions for $p \twoheadrightarrow q$, i.e.:

$$\vdash s(p \twoheadrightarrow q) \to (p \twoheadrightarrow q) \tag{31}$$
$$\vdash (p \twoheadrightarrow q) \to w(p \twoheadrightarrow q) \tag{32}$$

We now establish that s and w are, respectively, the *weakest* sufficient condition and the *strongest* necessary condition for $p \twoheadrightarrow q$ that can be expressed in IPC. Note that Lemma 8.10 below does not apply when p, q contain contractual implications.

Lemma 8.10. *Let p, q be IPC formulae. That is,*

1. If for an IPC formula c we have $\vdash c \to (p \twoheadrightarrow q)$, then $\vdash c \to q$.
2. If for an IPC formula c we have $\vdash (p \twoheadrightarrow q) \to c$, then $\vdash ((q \to p) \to q) \to c$.

Proof. For (1), by Lemma 8.7, we have $\vdash_{IPC} s(c \to (p \twoheadrightarrow q))$. That is $\vdash_{IPC} s(c) \to s(q)$, hence by Lemma 8.3 $\vdash_{IPC} c \to q$. We conclude by Lemma 8.4.

For (2), by Lemma 8.9, we have $\vdash_{IPC} w((p \twoheadrightarrow q) \to c)$. That is $\vdash_{IPC} ((w(q) \to w(p)) \to w(q)) \to w(c)$, hence by Lemma 8.3 $\vdash_{IPC} ((q \to p) \to q) \to c$. We conclude by Lemma 8.4. □

Here we justify the terms "strongest" and "weakest" for s and w.

Lemma 8.11. *Let $m \leq n$ be the preorder over* complete *homomorphic mappings m, n given by*

$$\vdash_{IPC} n(p \twoheadrightarrow q) \to m(p \twoheadrightarrow q) \text{ for all IPC formulas } p, q$$

Then, $s(\bullet)$ is a maximum and $w(\bullet)$ is a minimum. That is, $w \leq m \leq s$ for any complete m.

Proof. For s, we need to show that $\vdash_{IPC} s(p \twoheadrightarrow q) \to m(p \twoheadrightarrow q)$. By Lemma 8.3, $m(s(p \twoheadrightarrow q)) = s(p \twoheadrightarrow q)$, so we need to show $\vdash_{IPC} m(s(p \twoheadrightarrow q)) \to m(p \twoheadrightarrow q)$, which is $\vdash_{IPC} m(s(p \twoheadrightarrow q) \to (p \twoheadrightarrow q))$. By completeness, it is enough to show $\vdash s(p \twoheadrightarrow q) \to (p \twoheadrightarrow q)$, which is (31).

Similarly, for w we need to show that $\vdash_{IPC} m(p \twoheadrightarrow q) \to w(p \twoheadrightarrow q)$. By Lemma 8.3, $m(w(p \twoheadrightarrow q)) = w(p \twoheadrightarrow q)$, so we need to show $\vdash_{IPC} m(p \twoheadrightarrow q) \to m(w(p \twoheadrightarrow q))$, which is $\vdash_{IPC} m((p \twoheadrightarrow q) \to w(p \twoheadrightarrow q))$. By completeness, it is enough to show $\vdash (p \twoheadrightarrow q) \to w(p \twoheadrightarrow q)$, which is (32). □

We proved that s and w are complete homomorphic mappings. Several others do exist, however. Below, we provide a short table of those known to us at the time of writing.

$m(p \twoheadrightarrow q) = m(q)$ (s)
$m(p \twoheadrightarrow q) = (m(q) \to m(p)) \to m(q)$ (w)
$m(p \twoheadrightarrow q) = \neg\neg(m(q) \to m(p)) \to m(q)$
$m(p \twoheadrightarrow q) = \neg(m(q) \to m(p)) \vee m(q)$
$m(p \twoheadrightarrow q) = ((m(q) \to m(p)) \vee \mathsf{a}) \to m(q)$ for any prime a

The completeness proofs for these mappings are similar to those of Lemma 8.7 and 8.9, so we omit them. However no mapping can be sound, as we show below.

Theorem 8.12. *There is no sound homomorphic mapping from* PCL *to IPC.*

Proof. By contradiction, suppose there exists an homomorphic mapping $m(\bullet)$ such that $\vdash_{IPC} m(p) \implies \vdash p$ for all p. Take $p = m(\mathsf{q} \twoheadrightarrow \mathsf{r}) \leftrightarrow (\mathsf{q} \twoheadrightarrow \mathsf{r})$ for some prime q, r. Then $m(p) = m(m(\mathsf{q} \twoheadrightarrow \mathsf{r})) \leftrightarrow m(\mathsf{q} \twoheadrightarrow \mathsf{r}) = m(\mathsf{q} \twoheadrightarrow \mathsf{r}) \leftrightarrow m(\mathsf{q} \twoheadrightarrow \mathsf{r})$ by Lemma 8.2. The last form is trivially provable in IPC, so we can state $\vdash_{IPC} m(p)$. By the soundness of $m(\bullet)$, we have $\vdash p$. Hence

$$\vdash m(\mathsf{q} \twoheadrightarrow \mathsf{r}) \to (\mathsf{q} \twoheadrightarrow \mathsf{r})$$
$$\vdash (\mathsf{q} \twoheadrightarrow \mathsf{r}) \to m(\mathsf{q} \twoheadrightarrow \mathsf{r})$$

By Lemma 8.10, we have

$$\vdash m(\mathsf{q} \twoheadrightarrow \mathsf{r}) \to \mathsf{r}$$
$$\vdash ((\mathsf{r} \to \mathsf{q}) \to \mathsf{r}) \to m(\mathsf{q} \twoheadrightarrow \mathsf{r})$$

hence, $\vdash ((\mathsf{r} \to \mathsf{q}) \to \mathsf{r}) \to \mathsf{r}$. By Lemma 8.5 we have $\vdash_{IPC} ((\mathsf{r} \to \mathsf{q}) \to \mathsf{r}) \to \mathsf{r}$. This is however false, since IPC can not prove Peirce's law (not even on prime formulae). □

8.2 Mappings to S4

We shall consider the extensions of a standard IPC mapping to S4 [20, 17].

Definition 8.13. An extended mapping to S4 is a function $m(\bullet)$ from PCL to S4 such that

$$
\begin{aligned}
m(\mathsf{p}) &= \Box\mathsf{p} \\
m(p \wedge q) &= m(p) \wedge m(q) \\
m(p \vee q) &= m(p) \vee m(q) \\
m(p \to q) &= \Box(m(p) \to m(q)) \\
m(\top) &= \top \\
m(\bot) &= \bot \\
m(\neg p) &= \Box\neg m(p) \\
m(p \twoheadrightarrow q) &= \mathcal{C}[m(p), m(q)]
\end{aligned}
$$

for some fixed S4 formula context $\mathcal{C}$.

We now introduce some technical lemmas.

Definition 8.14. A formula p is $\Box$-invariant iff $\vdash_{S4} p \leftrightarrow \Box p$.

Note that the $\leftarrow$ direction is actually true for all p.

Lemma 8.15. *If p, q are $\Box$-invariant, then $m(p \wedge q), m(p \vee q), m(p \to q), m(\neg p), m(\top), m(\bot), m(\mathsf{a})$ (with a prime) are such.*

Proof. The cases $\top, \bot$ are trivial. For the others, we proceed as follows. For $\wedge$, we simply apply $\vdash_{S4} \Box a \wedge \Box b \to \Box(a \wedge b)$. Similarly, for $\vee$, we apply $\vdash_{S4} \Box a \vee \Box b \to \Box(a \vee b)$. For $\neg, \to, \mathsf{a}$, we apply $\vdash_{S4} \Box a \to \Box\Box a$. □

Lemma 8.16. *Assume that, for all $\Box$-invariant p, q, $m(p \twoheadrightarrow q)$ is $\Box$-invariant. Then, for any p, $m(p)$ is $\Box$-invariant.*

Proof. By structural induction on p. Indeed, all the inductive steps are covered by either the hypothesis or Lemma 8.15. □

Lemma 8.17. *If a is prime, $m(q)$ is $\Box$-invariant, and $\vdash_{S4} m(p)$, then we have $\vdash_{S4} m(p\{q/\mathsf{a}\})$.*

Proof. Clearly, $m(p) = \mathcal{C}[m(\mathsf{a})] = \mathcal{C}[\Box\mathsf{a}]$ for some context $\mathcal{C}$. So, for the same context, $m(p\{q/\mathsf{a}\}) = \mathcal{C}[m(q)]$. Since $m(q)$ is $\Box$-invariant, the latter is equivalent in S4 to $\mathcal{C}[\Box m(q)] = \mathcal{C}[\Box\mathsf{a}]\{m(q)/\mathsf{a}\} = m(p)\{m(q)/\mathsf{a}\}$. The last formula holds in S4, by substituting a in $\vdash_{S4} m(p)$. □

We are aware of several complete but unsound mappings to S4.

Definition 8.18. The extended mappings $e_1, \ldots, e_4$ to S4 are defined as follows:

$$
\begin{aligned}
e_1(p \twoheadrightarrow q) &= (e_1(q) \to e_1(p)) \to e_1(q) \\
e_2(p \twoheadrightarrow q) &= \Diamond(\Diamond e_2(q) \to e_2(p)) \to e_2(q) \\
e_3(p \twoheadrightarrow q) &= \Box(\Box(e_3(q) \to e_3(p)) \to e_3(q)) \\
e_4(p \twoheadrightarrow q) &= \Box(\Box(e_4(q) \to \Diamond e_4(p)) \to e_4(q))
\end{aligned}
$$

Lemma 8.19. *$e_i(p)$ is $\Box$-invariant.*

Proof. By Lemma 8.16, we only need to check that $e_i(p \twoheadrightarrow q)$ is $\Box$-invariant whenever p, q are such. If we write $\mathcal{C}_i$ for the contexts such that $e_i(p \twoheadrightarrow q) = \mathcal{C}_i[e_i(p), e_i(q)]$, then it is sufficient to verify that

$$\vdash_{S4} (\mathsf{a} \leftrightarrow \Box\mathsf{a}) \wedge (\mathsf{b} \leftrightarrow \Box\mathsf{b}) \rightarrow (\mathcal{C}_i[\mathsf{a}, \mathsf{b}] \leftrightarrow \mathcal{C}_i[\Box\mathsf{a}, \Box\mathsf{b}]) \tag{33}$$

and then conclude by substituting the prime formulae a, b. Formula (33), for each i, can be easily verified. A simple way to do it is to use a S4 theorem prover. We used LWB for this: see Sect. A.3 for more details. □

Lemma 8.20. $\vdash_{S4} e_i(\mathsf{p} \twoheadrightarrow \mathsf{q}) \leftrightarrow e_j(\mathsf{p} \twoheadrightarrow \mathsf{q})$ *iff* $i = j$.

Proof. This is an easy, albeit long, exercise. See Sect. A.3 for more details. □

Lemma 8.21. *The mappings $e_i(\bullet)$ are complete, i.e.* $\vdash p \implies \vdash_{S4} e_i(p)$*, for $i \in [1..4]$. The converse is false, in general, so they are unsound.*

Proof. We proceed by induction on the derivation of $\vdash p$.

If p is an instance of a PCL axiom, by Lemma 8.16 and 8.17 it is sufficient to consider the case of the axiom being applied to *prime* distinct formulae, since we can then substitute them to obtain p. Therefore, we check whether $\vdash e_i(q)$ for each prime instance of each PCL axiom q. This generates a finite number of specific formulae to verify in S4. Since this is rather long, we resort to LWB for this task. See Sect. A.3 for more details.

If instead $\vdash p$ was derived through a cut rule, say $\dfrac{a \quad a \rightarrow p}{p}$, then, by the inductive hypothesis, we have $\vdash_{S4} m(a)$ and $\vdash_{S4} m(a \rightarrow p) = \Box(m(a) \rightarrow m(p))$. Since $\Box q \rightarrow q$ is a tautology of S4, we can have $\vdash_{S4} m(a) \rightarrow m(p)$. By the cut rule of S4, we conclude $\vdash_{S4} m(p)$.

For the unsoundness result, it is enough to check that $\vdash_{S4} e_i(p_i)$ for some p_i such that $\nvdash p_i$. This was checked using the LWB theorem prover: we refer to Sect. A.3 for more details. □

8.3 Lax PCL

Propositional lax logic (PLL) [15] is an extension of IPC with a single modality $\bigcirc$, called *lax modality*, characterized by the following axioms:

$p \rightarrow \bigcirc p$	$\bigcirc\mathsf{R}$
$\bigcirc \bigcirc p \rightarrow \bigcirc p$	$\bigcirc\mathsf{M}$
$(p \rightarrow q) \rightarrow (\bigcirc p \rightarrow \bigcirc q)$	$\bigcirc\mathsf{F}$

These axioms appear to be relevant for contracts. Suppose we have a contract at hand, and reason about its implications. If we read $\bigcirc p$ as "p is ensured by the contract", then the axioms agree with our intuition. Axiom $\bigcirc\mathsf{R}$ states that true propositions are always guaranteed. Axiom $\bigcirc\mathsf{M}$ states that committing to a promise is actually a promise itself. Axiom $\bigcirc\mathsf{F}$ is simple: if p is ensured, and implies q, clearly q must be ensured as well.

One might expect that, if we take a fixed formula c expressing the requirements of a contract, and we interpret $\bigcirc q$ as $c \twoheadrightarrow q$, then this should satisfy the axioms above. In other words, we expect $\bigcirc = c \twoheadrightarrow \bullet$ to be a lax modality (for any fixed c). However, it turns out that this is not the case in PCL.

Lemma 8.22. PCL *proves* $\bigcirc$R *and* $\bigcirc$F*, but not* $\bigcirc$M*.*

Proof. Under the above definition of $\bigcirc$, Axiom $\bigcirc$R holds, as one can derive it using PrePost and Zero. Axiom $\bigcirc$F is also a special case of PrePost. Axiom $\bigcirc$M does not hold, that is $\not\vdash (c \twoheadrightarrow (c \twoheadrightarrow p)) \rightarrow (c \twoheadrightarrow p)$: this can be verified through the decision procedure of Lemma 6.14. □

Quite interestingly, sometimes PLL is axiomatized in an alternative equivalent way. Instead of including $\bigcirc$F as an axiom, a related inference rule is used, together with another axiom:

$$\frac{p \rightarrow q}{\bigcirc p \rightarrow \bigcirc q} \qquad \bigcirc\mathsf{F}$$

$$(\bigcirc q \wedge \bigcirc r) \rightarrow \bigcirc(q \wedge r) \qquad \bigcirc\mathsf{S}$$

Axiom $\bigcirc$S does not hold either in PCL under our interpretation for $\bigcirc$. We actually already stated this when we discussed property (21). We will not use this alternative axiomatization, since it uses another inference rule, and we find this inconvenient.

We now discuss how to extend PCL so that the indexed modality $\bigcirc_p$, defined as $\bigcirc_p q = p \twoheadrightarrow q$, is a lax modality. We need to adapt both our Hilbert-style axiomatization and our sequent calculus rules. For the Hilbert-style proof system, we simply augment the axioms of Def. 4.2 with the $\bigcirc$M axiom, following Lemma 8.22. We call PCL^{Lax} the logic extended as such.

Definition 8.23. The Hilbert-style axiomatisation of PCL^{Lax} extends that of PCL (Def. 4.2) with the following axiom:

$$(p \twoheadrightarrow (p \twoheadrightarrow q)) \rightarrow (p \twoheadrightarrow q) \qquad \mathsf{Lax}$$

In this section we redefine $\vdash$ to stand for provability in PCL^{Lax}.

We now adapt the sequent calculus of Def. 6.1.

Definition 8.24. The sequent calculus for PCL^{Lax} extends that of PCL (Def. 6.1) with the rule:

$$\frac{\Gamma, p \twoheadrightarrow q, a \vdash p \quad \Gamma, p \twoheadrightarrow q, q \vdash a \twoheadrightarrow b}{\Gamma, p \twoheadrightarrow q \vdash a \twoheadrightarrow b}\mathsf{Lax}$$

The above rule is best understood when compared with PrePost:

$$\frac{\Gamma, p \twoheadrightarrow q, a \vdash p \quad \Gamma, p \twoheadrightarrow q, q \vdash b}{\Gamma, p \twoheadrightarrow q \vdash a \twoheadrightarrow b}\mathsf{PrePost}$$

The single difference between PrePost and Lax is the use of b instead of $a \twoheadrightarrow b$ in the second premise. Indeed, because of this, rule Lax is more general, in that it subsumes rule PrePost.

Lemma 8.25. *The* PrePost *rule is redundant in the* PCL^{Lax} *sequent calculus.*

Proof. Indeed, we can replace each use of PrePost in a derivation using Lax as follows:

$$\dfrac{\dfrac{D0}{\Gamma, p \twoheadrightarrow q, a \vdash p} \quad \dfrac{D1}{\Gamma, p \twoheadrightarrow q, q \vdash b}}{\Gamma, p \twoheadrightarrow q \vdash a \twoheadrightarrow b}\mathsf{PrePost} \quad \Longrightarrow$$

$$\dfrac{\dfrac{D0}{\Gamma, p \twoheadrightarrow q, a \vdash p} \quad \dfrac{\dfrac{D1}{\Gamma, p \twoheadrightarrow q, q \vdash b}}{\Gamma, p \twoheadrightarrow q, q \vdash a \twoheadrightarrow b}\mathsf{Zero}}{\Gamma, p \twoheadrightarrow q \vdash a \twoheadrightarrow b}\mathsf{Lax}$$

□

Consequently, we will neglect rule PrePost from now on. We can now restate a number of fundamental results, starting from the Hilbert-Gentzen equivalence.

Theorem 8.26. $\vdash_H p \iff \vdash_G p$

Proof. We adapt the proof of Th.6.4. To prove $\vdash_H p \implies \vdash_G p$, we just need to derive the Lax axiom in the sequent calculus.

$$\dfrac{\dfrac{\dfrac{}{p \twoheadrightarrow (p \twoheadrightarrow q), p \vdash p}\mathsf{id} \quad \dfrac{}{p \twoheadrightarrow (p \twoheadrightarrow q), p \twoheadrightarrow q \vdash p \twoheadrightarrow q}\mathsf{id}}{p \twoheadrightarrow (p \twoheadrightarrow q) \vdash p \twoheadrightarrow q}\mathsf{Lax}}{\vdash (p \twoheadrightarrow (p \twoheadrightarrow q)) \to (p \twoheadrightarrow q)} \to \mathsf{R}$$

For the other direction, $\vdash_G p \implies \vdash_H p$, we provide the missing case:

- Rule Lax: $[(\bigwedge \Gamma \wedge (p \twoheadrightarrow q) \wedge a) \to p] \wedge [(\bigwedge \Gamma \wedge (p \twoheadrightarrow q) \wedge q) \to (a \twoheadrightarrow b)] \to (\bigwedge \Gamma \wedge (p \twoheadrightarrow q)) \to (a \twoheadrightarrow b)$.
  Assume all the hypotheses. We get $\Gamma, p \twoheadrightarrow q$, hence $a \to p, q \to (a \twoheadrightarrow b)$. By PrePost, we get $p \twoheadrightarrow (a \twoheadrightarrow b)$. Again by PrePost, we get $a \twoheadrightarrow (a \twoheadrightarrow b)$. We apply Lax to conclude $a \twoheadrightarrow b$.

□

Lemma 8.27. *The* Lax *rule is not redundant.*

Proof. Otherwise, the PrePost rule would be redundant in plain PCL, so we conclude by Lemma 6.15 and Lemma 8.25. □

Theorem 8.28. (Cut-elimination) *The* cut *rule is redundant in* PCL^{Lax}.

Proof. We proceed as in Sect.7, replacing the PrePost rule with the Lax one. Cases not involving PrePost are thus unaffected. According to Table 1, we need to check four new cases: three essential (Zero/Lax, Lax/Lax, Lax/Fix), and one right commutation (*/Lax). We now define the reduction relation $\Longrightarrow$ for these cases.

- Case Zero/Lax (essential)

$$\dfrac{\dfrac{D0}{\Gamma \vdash q}\mathsf{Zero} \quad \dfrac{D1 \qquad D2}{\Gamma, p \twoheadrightarrow q, a \vdash p \qquad \Gamma, p \twoheadrightarrow q, q \vdash a \twoheadrightarrow b}\mathsf{Lax}}{\Gamma \vdash a \twoheadrightarrow b}\mathsf{cut} \quad \Longrightarrow$$

$$\dfrac{D0 \qquad \dfrac{\dfrac{}{\Gamma, q \vdash q}\mathsf{id}}{\Gamma, q \vdash p \twoheadrightarrow q}\mathsf{Zero} \quad D2}{\Gamma \vdash a \twoheadrightarrow b}\mathsf{cut_1}}{\Gamma \vdash q \qquad \Gamma, q \vdash a \twoheadrightarrow b}\mathsf{cut_p}$$

- Case Lax/Lax (essential)

$$\dfrac{\dfrac{D0 \qquad D1}{\Gamma, p0 \vdash p \qquad \Gamma, q \vdash p0 \twoheadrightarrow q0}\mathsf{Lax} \quad \dfrac{D2 \qquad D3}{\Delta, a0 \vdash p0 \qquad \Delta, q0 \vdash a0 \twoheadrightarrow b0}\mathsf{Lax}}{\Gamma \vdash p0 \twoheadrightarrow q0 \qquad \Delta = \Gamma, p0 \twoheadrightarrow q0 \vdash a0 \twoheadrightarrow b0}\mathsf{cut} \quad \Longrightarrow$$

$$\dfrac{Da \quad Db}{\Gamma(p \twoheadrightarrow q) \vdash a0 \twoheadrightarrow b0}\mathsf{Lax}$$

$$Da = \dfrac{\dfrac{\dfrac{D0+}{\Gamma, a0, p0 \vdash p} \quad \dfrac{D1+}{\Gamma, a0, q \vdash p0 \twoheadrightarrow q0}\mathsf{Lax}}{\Gamma, a0 \vdash p0 \twoheadrightarrow q0} \quad \dfrac{D2+}{\Gamma, a0, p0 \twoheadrightarrow q0 \vdash p0}\mathsf{cut_1}}{\Gamma, a0 \vdash p0} \quad \dfrac{D0+}{\Gamma, a0, p0 \vdash p}\mathsf{cut_p}}{\Gamma, a0 \vdash p}$$

$$Db = \dfrac{\dfrac{D1}{\Gamma, q \vdash p0 \twoheadrightarrow q0} \quad \dfrac{D2+ \qquad D3+}{\Gamma, q, p0 \twoheadrightarrow q0, a0 \vdash p0 \qquad \Gamma, q, p0 \twoheadrightarrow q0, q0 \vdash a0 \twoheadrightarrow b0}\mathsf{Lax}}{\Gamma, q \vdash a0 \twoheadrightarrow b0}\mathsf{cut_0}$$

- Case Lax/Fix (essential)

$$\dfrac{\dfrac{D0}{\Gamma, p0 \vdash p} \quad \dfrac{D1}{\Gamma, q \vdash p0 \twoheadrightarrow q0}\mathsf{Lax} \quad \dfrac{D2 \qquad D3}{\Gamma, r, p0 \twoheadrightarrow q0 \vdash p0 \qquad \Gamma, q0, p0 \twoheadrightarrow q0 \vdash r}\mathsf{Fix}}{\Gamma(p \twoheadrightarrow q) \vdash r}\mathsf{cut} \quad \Longrightarrow$$

$$\dfrac{Da \quad Db}{\Gamma(p \twoheadrightarrow q) \vdash r}\mathsf{Fix}$$

$$Da = \dfrac{\dfrac{}{\Gamma, r, p \vdash p}\mathsf{id} \quad \dfrac{\dfrac{\dfrac{D1+}{\Gamma, r, q \vdash p0 \twoheadrightarrow q0} \quad \dfrac{D2+}{\Gamma, r, q, p0 \twoheadrightarrow q0 \vdash p0}\mathsf{cut_0}}{\Gamma, r, q \vdash p0} \quad \dfrac{D0+}{\Gamma, r, q, p0 \vdash p}\mathsf{cut_p}}{\Gamma, r, q \vdash p}}{\Gamma, r \vdash p}\mathsf{Fix}$$

$$Db = \dfrac{\dfrac{D1}{\Gamma, q \vdash p0 \twoheadrightarrow q0} \quad \dfrac{D2+ \qquad D3+}{\Gamma, q, p0 \twoheadrightarrow q0, r \vdash p0 \qquad \Gamma, q, p0 \twoheadrightarrow q0, q0 \vdash r}\mathsf{Fix}}{\Gamma, q \vdash r}\mathsf{cut_0}$$

- Case */Lax (right commutation)

$$\dfrac{\dfrac{\dfrac{D0}{\Gamma \vdash r} \quad \dfrac{\dfrac{D1}{\Gamma, r, a \vdash p} \quad \dfrac{D2}{\Gamma, r, q \vdash a \twoheadrightarrow b}}{\Gamma, r \vdash a \twoheadrightarrow b}\mathsf{Lax}}{\Gamma(p \twoheadrightarrow q) \vdash a \twoheadrightarrow b}\mathsf{cut} \quad \Longrightarrow$$

$$\dfrac{\dfrac{\dfrac{D0+}{\Gamma, a \vdash r} \quad \dfrac{D1}{\Gamma, a, r \vdash p}}{\Gamma, a \vdash p}\mathsf{cut_1} \quad \dfrac{\dfrac{D0+}{\Gamma, q \vdash r} \quad \dfrac{D2}{\Gamma, r, q \vdash a \twoheadrightarrow b}}{\Gamma, q \vdash a \twoheadrightarrow b}\mathsf{cut_1}}{\Gamma(p \twoheadrightarrow q) \vdash a \twoheadrightarrow b}\mathsf{Lax}$$

□

Of course, we are able to restate decidability.

Lemma 8.29. *The sequent calculus of* PCL^{Lax} *satisfies the subformula property.*

Proof. Trivial inspection of the rule Lax. □

Theorem 8.30. (Decidability) *The logic* PCL^{Lax} *is decidable.*

Proof. Immediate by Lemma 8.29 and Th.8.28. See Th.6.14 for more details. □

9 Principals

As illustrated by the examples in Sect. 5, PCL allows for inferring whether some promise is implied by a set of contracts. In real-world scenarios, each contract will be issued by a given *principal*. It would then be useful to represent the binding between principals and contracts within the logic. This would allow, for instance, to single out the principal who is responsible for a violation, and possibly take countermeasures against him.

To this aim, we extend our logic with a *says* modality, similarly to [17].

Definition 9.1. The syntax of PCL^{says} extends that of PCL (Def. 4.1) as follows:

$$p ::= \cdots \mid A \textit{ says } p$$

where we assume a set of (atomic) *principals*, ranged over by $A, B, \ldots$.

We can then rephrase Alice's contract from the toy exchange example in Sect. 1 as *Alice says* $(\mathsf{b} \twoheadrightarrow \mathsf{a})$. This represents the fact that the principal named Alice has issued a contract, where she promises to lend her airplane, provided that she borrows a bike.

We now develop the proof theory of PCL^{says}. Essentially, we extend the PCL axioms with those of the logic ICL [17]. This is an indexed lax logic, where the lax modality corresponds to our *says*. The interesting contribution is that this extension preserves all the main results of PCL, in particular its decidability.

Definition 9.2. The Hilbert-style axiomatisation of PCL^{says} extends that of PCL (Def. 4.2) with the following axioms:

$$p \rightarrow (A \textit{ says } p) \qquad \mathsf{SaysR}$$
$$(A \textit{ says } A \textit{ says } p) \rightarrow A \textit{ says } p \qquad \mathsf{SaysM}$$
$$(p \rightarrow q) \rightarrow (A \textit{ says } p) \rightarrow (A \textit{ says } q) \qquad \mathsf{SaysF}$$

Back to our toy exchange example of Sect. 1, one would expect to deduce an agreement between Alice and Bob, that commits Alice to lend her airplane (i.e., *Alice says* a), and commits Bob to lend his bike (i.e., *Bob says* b). A careful specification of the kids contracts is however needed, in order to obtain the expected result. As a first attempt, consider the following specification:

$$p_{toy} \;=\; \textit{Alice says } (\mathsf{b} \twoheadrightarrow \mathsf{a}) \;\wedge\; \textit{Bob says } (\mathsf{a} \twoheadrightarrow \mathsf{b})$$

While, at first sight, this specification seems enough to deduce the desired result, it turns out that a different (and quite weaker) handshaking is reached, i.e.:

$$p_{toy} \;\rightarrow\; \textit{Alice says Bob says } (\mathsf{a} \wedge \mathsf{b})$$

That is, Alice and Bob together promise to exchange their toys. However, this only tells half of the story: actually, it does not distinguish the duties of Alice from those of Bob. Actually, the conditions required by Alice and Bob (respectively, b and a) do not precisely capture our intuition. In the contract *Alice says* $(\mathsf{b} \twoheadrightarrow \mathsf{a})$, a promise of b from Bob is not enough to commit Alice to lend her airplane: requiring b means that Alice wants indeed to have the bike. As a result, in the specification p_{toy} we can deduce that a and b hold only under the responsibility of both Alice and Bob.

The specification can be made more precise as follows. Alice promises to lend her airplane, provided that Bob promises to lend his bike (and *vice versa*). The overall contract can then be rephrased as:

$$p'_{toy} \;=\; \textit{Alice says } ((\textit{Bob says } \mathsf{b}) \twoheadrightarrow \mathsf{a}) \;\wedge\; \textit{Bob says } ((\textit{Alice says } \mathsf{a}) \twoheadrightarrow b)$$

The new contract reveals the whole story, that is, the duty of Alice is that of lending her airplane, and the duty of Bob is that of lending his bike.

$$p'_{toy} \;\rightarrow\; \textit{Alice says } \mathsf{a} \;\wedge\; \textit{Bob says } \mathsf{b}$$

The above obligations can be exploited by a third party (a sort of "automated" judge) which has to investigate the responsibilities of the involved parties, in the unfortunate case that the contract is not respected. For instance, if our judge is given the evidence that Alice's airplane has never been lent to Bob, then he will be able to infer that Alice has not respected her contract (and possibly punish her), that is:

$$p'_{toy} \wedge \neg\mathsf{a} \;\rightarrow\; (\textit{Alice says } \mathsf{a} \wedge \neg\mathsf{a}) \;\rightarrow\; \textit{Alice says } \bot$$

Explicitly representing principals has some additional benefits, especially when putting our logic at work in insecure environments populated by attackers. Actually, an attacker could maliciously issue a "fake" contract, where he makes a promise that he cannot actually implement, e.g. because the promised duty can

only be performed by another party. By binding each contract with its principal, it is easy to realize when someone has attempted such a fraud, because the principal who has signed the contract is different from who is due to implement the promised behaviour.

Back to our technical development, the main results of PCL are also enjoyed by its extension PCL^{says} .

Definition 9.3. The sequent calclus of PCL^{says} includes all the rules for PCL, and the following additional rules:

$$\frac{\Gamma \vdash p}{\Gamma \vdash A \ says \ p}\mathsf{SaysR}$$

$$\frac{\Gamma,\ A \ says \ p,\ p \vdash A \ says \ q}{\Gamma, A \ says \ p \vdash A \ says \ q}\mathsf{SaysL}$$

Theorem 9.4. (Cut Elimination) *If p is provable in PCL^{says}, then there exists a proof of p which does not use the* Cut *rule.*

Proof. We use the structural approach of [27], similarly to Sect. 7. The cases involving only PCL rules have been already dealt with in Sect. 7. The cases involving only *says* are dealt with in [17]. The other cases are commutations, and so they are dealt with either as in Sect. 7 or as in [17]. □

Lemma 9.5. *The sequent calculus of PCL^{says} satisfies the subformula property.*

Proof. Trivial inspection of rules SaysR and SaysL. □

Theorem 9.6. (Decidability) *The logic PCL^{says} is decidable.*

Proof. Direct from Th.9.4 and Lemma 9.5. □

10 Related Work

Various approaches to the problem of providing both clients and services with provable guarantees about each other's functional behaviour have been studied over the last few years. Yet, at the present no widespread technology seems to give a general solution to this problem.

The motivations underlying our contract logic seem somewhat related to those for the logics introduced in [3] to study how to compose assume-guarantee specifications of concurrent systems [2]. The idea is that a system will give some guarantee M_1 about its behaviour, provided that also the environment it operates within will behave according to some assumption M_2, and *vice versa*. This is rendered in [3] as the intuitionistic formula $(M_1 \to M_2) \wedge (M_2 \to M_1)$. However, the technical development of the two approaches is quite different. In our approach, we use contractual implication $\twoheadrightarrow$, rather than implication $\to$, in order to obtain $M_1 \wedge M_2$ from $(M_1 \twoheadrightarrow M_2) \wedge (M_2 \twoheadrightarrow M_1)$. In [3], instead, the connective $\to$ is the usual implication of intuitionistic logic. Therefore, simply adding the axiom $(M_1 \to M_2) \wedge (M_2 \to M_1) \vdash M_1 \wedge M_2$ would make the logic inconsistent. To overcome this problem, in [3] the above judgement only

holds in particular models, which are subject to several constraints (e.g., the propositions M_1, M_2 must be interpreted as safety properties). While this allows for exploring these models at some depth, our approach appears more general, since it develops a proof theory of contracts while abstracting from the specific models of the logic.

The complexity of real-world scenarios, where several concepts like principals, contracts, authorizations, duties, delegation, mandates, regulations, *etc.* are inextricably intermingled, have led to a steady flourishing of new logics over the years. The logics proposed to model such scenarios take inspiration and extend e.g. classical [12], modal [11], deontic [28, 18], default [19] and defeasible logics [21]. We think none of these logics, including our PCL, captures *all* the facets of contracts. Each of these logics is designed to represent some particular aspect of contracts, e.g. obligations, permissions and prohibitions in deontic logics, violation of contracts in default and defeasible logics, and agreement in our contract logic. We argue that, since these aspects are orthogonal, it is possible to extend PCL with features from some of these logics.

Our research seems also related to foundational research on authorization logics for distributed systems [1, 17, 23]. There, the problem is that of deciding whether, given a bunch of authorization assertions modelled in the logic (possibly involving complex concepts like roles, groups, delegations), we can deduce that a principal has the right to access a given resource. While many of these concepts are common with contracts, there is a fundamental difference between the two worlds. While authorizations logics are focussed on deciding *if* a principal is allowed to perform some action, in our contract logic we are also concerned with discovering *what* that principal has to promise in return.

Recent research papers address the problem of defining contracts that specify the interaction patterns among (clients and) services [8, 9, 25]. For instance, in [9] a contract is a process in a process algebra featuring only prefixing, internal and external choice. A client contract is compliant with a service contract if any possible interaction between the client and the service will always succeed. There, a main problem is how to define (and decide) a subcontract relation, that allows for safely substituting services without affecting the compliance with their clients. Even assuming that services are trusted and respect the published contract, this approach provides the client with no provable guarantees, except that the interaction with the service will "succeed", that is all the expected synchronizations will take place. For instance, consider a simple buyer-seller scenario. In our vision, it is important to provide the buyer with the guarantee that, e.g., after the payment has been made, then either the payed goods are made available, or a full refund is issued. For the seller, it is important to be sure that, e.g., a buyer will not repudiate a completed transaction, so to obtain for free the goods already delivered. This could be modelled by the following contracts, assuming a perfect duality between buyer and seller:

$$\mathit{Buyer} = (\mathsf{ship} \vee \mathsf{refund}) \twoheadrightarrow \mathsf{pay} \qquad \mathit{Seller} = \mathsf{pay} \twoheadrightarrow (\mathsf{ship} \vee \mathsf{refund})$$

The above two contracts lead to an agreement, which allows the buyer for paying, and the seller for shipping or issuing a refund. Instead, in [9] the contracts of the buyer and of the seller would take a very different form, e.g.:

$$\mathit{Buyer} = \overline{\mathsf{pay}}.\,(\mathsf{ship} + \mathsf{refund}) \qquad \mathit{Seller} = \mathsf{pay}.\,(\overline{\mathsf{ship}} \oplus \overline{\mathsf{refund}})$$

Intuitively, this means that the client will first output a payment, and then either receive the item, or receive a refund (at service discretion). Dually, the service will first input a payment, and then opt for shipping the item or issuing a refund. However, this is very distant from our notion of contracts.

First, such contracts are quite rigid, in that they precisely fix the order in which the actions must be performed. Even though in some cases this may be desirable, many real-world contracts seem to allow for a more liberal way of constraining the involved parties (e.g., "I will pay before the deadline").

Second, while the crucial notion if the contracts in [9] is *compatibility*, in our model we focus on the inferring the *obligations* that arise from a set of contracts. The key difference between the two notions is that, given a set of contracts, a compatibility check results in a yes/no output, while inferring the obligations provides a fine-grained quantification of the reached agreement. For instance, the obligations may identify who is responsible of each action mentioned in the contract. We can then exploit this information to take some recovery action against clients and services which do not respect their promises.

A lot of work addresses the problem of managing service failures in long-running business transactions, see e.g. [10, 5, 7, 6]. Since, in a long-lived transaction, the standard rollback mechanism of database systems does not scale, the idea is to partition the long transaction into a sequence of smaller transactions, each of which is associated with a given compensation [16]. Compensations are recovery actions specified by the service designer, that will be run upon failures of the standard execution. For instance, we can model our buyer-seller scenario as follows in Compensating CSP [7]. The seller charges the buyer credit card, and then proceeds by shipping the ordered item. Simultaneously, the seller performs an availability check, to see whether the ordered item is in stock. If the item is not available, the service throws an exception, which triggers the compensation refundAmount. This compensation restores the original state of the buyer account, so it will actually perform a transaction rollback:

$$\begin{aligned} \mathit{Seller} \;=\; & [\; \mathsf{availCheck}; (\mathsf{ok}; \mathit{SKIPP} \;\Box\; \mathsf{notOk}; \mathit{THROWW}) \\ & \quad \|\; (\mathsf{debitAmount} \div \mathsf{refundAmount})\;]; \mathsf{ship} \end{aligned}$$

Notice that the choice of the compensation is crucial; while "refundAmount" may be acceptable by any client, if the compensation was instead just a "15%Discount" on the next order, then not all the clients would have been perfectly happy. Actually, our main criticism to long-running transactions is that clients have absolutely no control on the compensations provided by services.

In our vision, instead, clients have the right to select those services that offer the desired compensations. For instance, we may exploit our logic to model the contract of a buyer that will pay provided that, if the ordered item is unavailable, then she will obtain a full refund, as well as a 15% discount on the next order:

$$\mathit{Buyer} \;=\; \big(\mathsf{unavailable} \rightarrow (\mathsf{refund} \wedge 15\%\mathsf{discount})\big) \twoheadrightarrow \mathsf{pay}$$

11 Conclusions and Future Work

We have investigated the notion of contract from a logical perspective. To do that, we have extended intuitionistic propositional logic with a new connective, that models contractual implication. We have provided the new connective with

an Hilbert-style axiomatisation, which have allowed us to deduce, for instance, that n contracting parties, each requiring a promise from the other $n-1$ parties in order to make its own promise, eventually reach an agreement. Further interesting properties and application scenarios for our logic have been explored, in particular in Sect. 3 and 5.

The main result about our logic is its decidability. To prove that, we have devised a Gentzen-style sequent calculus for the logic, which is equivalent to the Hilbert-style axiomatisation. Decidability then follows from the subformula property, which is enjoyed by our Gentzen rules, and by a cut elimination theorem, which we have proved in full details in this paper. As a further support to our logic, we have implemented a proof search algorithm, which decides if any given formula is a tautology or not.

Our logic for contracts has served as a basic building block for designing a calculus of contracting processes [4]. This is an extension of Concurrent Constraints, featuring a peculiar mechanism for the fusion of variables, which well suites to formalise contract agreements. Our calculus is expressive enough to model a variety of typical scenarios, and to encode some common idioms for concurrency, among which the π-calculus and graph rewriting.

11.1 Future Work

While designing the logic for contracts proposed in this paper, our main concerns were to give a minimal set of rules for capturing the notion of contract agreement, and at the same time preserving the decidability of IPC.

We expect that many useful features can be added to our logic, to make it suitable for modelling complex scenarios, which are not directly manageable with the basic primitives presented here. Of course, preserving the decidability of the logic will be a major concern, while considering these extensions. We discuss below some of the additional features which we think to be more useful in the future developments of our logic.

First order features. A significant extension to our logic would be that of extending it with predicates and quantifiers. This will allow us to model more accurately several scenarios, where a party issues a "generic" contract that can be matched by many parties. While this first order extension shall force us to drop the decidability result, we expect to find interesting decidable fragments of the logic, through which modelling many relevant situations.

For instance, consider an e-commerce scenario, where a seller promises to ship the purchased item to a given address, provided that the customer will pay for that item. Aiming at generality, we make the seller contract parametric with respect to the item, customer and address. This could be modelled using a universal quantification over these three formal parameters:

$$\begin{aligned} \mathit{Seller} = \forall \mathit{item}, \mathit{customer}, \mathit{address} : \\ \mathsf{pay}(\mathit{item}, \mathit{customer}, \mathit{address}) \twoheadrightarrow \mathsf{ship}(\mathit{item}, \mathit{address}) \end{aligned} \tag{34}$$

Now, assume that a customer (say, Bob) promises that he will pay for a drill, provided that the seller will ship the item to his address. This is modelled by the following contract issued by Bob, where the actual parameters remark that

the actual payment is made by Bob, and that the destination address is Bob's.

$$Bob = \mathsf{ship}(\mathsf{drill}, \mathsf{bobAddress}) \twoheadrightarrow \mathsf{pay}(\mathsf{drill}, \mathsf{Bob}, \mathsf{bobAddress})$$

Joining the two two contracts above will yield the intended agreement, that is:

$$Seller \wedge Bob \;\rightarrow\; \mathsf{pay}(\mathsf{drill}, \mathsf{Bob}, \mathsf{bobAddress}) \wedge \mathsf{ship}(\mathsf{drill}, \mathsf{bobAddress})$$

Consider now an attacker wanting to maliciously exploit the seller contract, in order to receive a free item, and to make the unaware customer Bob pay for it. To do that, the attacker issues the following contract:

$$\begin{aligned} FakeBob \;&= \mathsf{ship}(\mathsf{10Kdiamond}, \mathsf{fakeAddress}) \\ &\twoheadrightarrow \mathsf{pay}(\mathsf{10Kdiamond}, \mathsf{Bob}, \mathsf{fakeAddress}) \end{aligned}$$

Joining the seller and the attacker contracts will then cause an unwelcome situation for Bob, who is due to pay for a 10K diamond, which will be shipped to the attacker's address:

$$Seller \wedge FakeBob \;\rightarrow\; \begin{array}{l} \mathsf{pay}(\mathsf{10Kdiamond}, \mathsf{Bob}, \mathsf{fakeAddress}) \;\wedge \\ \mathsf{ship}(\mathsf{10Kdiamond}, \mathsf{fakeAddress}) \end{array}$$

To cope with this situation, we could require that each contract p is signed by the principal A who issues it, i.e. it has the form A *says* p. Revisiting our example with this trick, in the safe case that Bob himself has ordered the item, we would expect to deduce:

$$Seller \wedge Bob \;\rightarrow\; Bob \; says \; \mathsf{pay}(\mathsf{drill}, \mathsf{Bob}, \mathsf{bobAddress})$$

In this case, we have a successful transaction, because Bob is stating that he will pay for his drill. Instead, joining the seller and the attacker contracts produces:

$$\begin{array}{l} Seller \wedge FakeBob \;\rightarrow \\ FakeBob \; says \; \mathsf{pay}(\mathsf{10Kdiamond}, \mathsf{Bob}, \mathsf{fakeAddress}) \end{array}$$

Now, it is easy to realize that someone has attempted a fraud, because the principal who has signed the contract (FakeBob) is different from that who is due to pay (Bob).

Explicit time. Time is another useful feature that may arise while modelling real-world scenarios. For instance, in an e-commerce transaction, a contract may state that if the customer returns the purchased item within 10 days from the purchase date, then she will have a full refund within 21 days from then.

We would like to model such a contract in a temporal extension of our logic, so to reason about the obligations that arise when the deadlines expire. Back to our e-commerce example, we could imagine to express the seller's contract as the following formula, where the parameter t in $\mathsf{p}(t)$ tells the point in time where the "event" p occurs:

$$Seller(t): \;\; \forall t' : (\mathsf{pay}(t) \wedge \mathsf{return}(t') \wedge t' < t+10) \twoheadrightarrow \exists t'' < t'+21 : \mathsf{refund}(t'')$$

From the point of view of the buyer, the contract says that the buyer is willing to pay, provided that she can obtain a full refund (within 21 days from the

date of payment), whenever she returns the item within 7 days from the date of payment:

$$Buyer(t): \quad \forall t' : (\mathsf{return}(t') \wedge t' < t+10 \rightarrow \exists t'' < t'+21 : \mathsf{refund}(t'')) \twoheadrightarrow \mathsf{pay}(t)$$

We expect our extended logic able to deduce that, in the presence of an agreement (i.e. a completed e-commerce transaction) between the customer and the seller on (say) January the 1st, 2009, if the customer has returned the purchased item on January the 5th, then the seller is required to issue a full refund to the customer within January, the 26th. This could be modelled by the formula:

$$Buyer(1.1.09) \wedge Seller(1.1.09) \wedge \mathsf{return}(5.1.09) \rightarrow \mathsf{refund}(26.1.09)$$

There are a number of techniques aimed at the explicit representation of time in logical systems, so we expect to be able to reuse some of them for extending PCL. These techniques range from Temporal Logic [14], to more recent approaches on temporal extensions of authorization logics like [13].

Process calculi and contracts. In this paper we have focussed our attention on logics-based formalisms for modelling contracts, and for deciding when they lead to an agreement among the involved parties. However, our investigation on contracts is still at its beginnings, and in future work we plan to study, along with logics for contracts, programming languages that exploit their features. As a first attempt towards this direction, we have designed in [4] a core calculus for contract-based computing.

In the future, we will continue to develop process calculi for contracts, in particular to describe the behaviour of processes in the presence of attackers. All the extensions we shall consider will have to preserve some characterizing features, e.g. the ability of publishing and stipulating contracts, that of deciding whether a given formula is on duty, and that of taking recovery actions in the case a contract is not respected.

We plan to develop analysis techniques to formally and automatically prove the correctness of the service infrastructure, e.g. that the contracts are always respected, without the need for resorting to third parties (e.g. legal offices) external to the model.

Acknowledgements

This work has been partially supported by EU-FETPI Global Computing Project IST-2005-16004 SENSORIA (Software Engineering for Service-Oriented Overlay Computers) and by the MIUR-PRIN project SOFT (Tecniche Formali Orientate alla Sicurezza).

References

[1] Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 4(15):706–734, 1993.

[2] Martín Abadi and Leslie Lamport. Composing specifications. *ACM Transactions on Programming Languages and Systems*, 15(1):73–132, 1993.

[3] Martín Abadi and Gordon D. Plotkin. A logical view of composition. *Theoretical Computer Science*, 114(1):3–30, 1993.

[4] Massimo Bartoletti and Roberto Zunino. A calculus of contracting processes. Technical Report DISI-09-056, DISI - Università di Trento, 2009.

[5] Laura Bocchi, Cosimo Laneve, and Gianluigi Zavattaro. A calculus for long running transactions. In *Proc. FMOODS*, 2003.

[6] Roberto Bruni, Hernán C. Melgratti, and Ugo Montanari. Theoretical foundations for compensations in flow composition languages. In *Proc. POPL*, 2005.

[7] Michael J. Butler, C. A. R. Hoare, and Carla Ferreira. A trace semantics for long-running transactions. In *25 Years Communicating Sequential Processes*, 2004.

[8] Samuele Carpineti and Cosimo Laneve. A basic contract language for web services. In *Proc. ESOP*, 2006.

[9] Giuseppe Castagna, Nils Gesbert, and Luca Padovani. A theory of contracts for web services. *ACM Transactions on Programming Languages and Systems*, 31(5), 2009.

[10] Mandy Chessell, Catherine Griffin, David Vines, Michael J. Butler, Carla Ferreira, and Peter Henderson. Extending the concept of transaction compensation. *IBM Systems Journal*, 41(4):743–758, 2002.

[11] Aspassia Daskalopulu and Tom Maibaum. Towards electronic contract performance. In *Proc. 12th International Workshop on Database and Expert Systems Applications*, 2001.

[12] Hasan Davulcu, Michael Kifer, and I.V. Ramakrishnan. CTR-S: A logic for specifying contracts in semantic web services. In *Proc. WWW04*, 2004.

[13] Henry DeYoung, Deepak Garg, and Frank Pfenning. An authorization logic with explicit time. In *Proc. CSF*, 2008.

[14] E. Allen Emerson. Temporal and modal logic. In *Handbook of Theoretical Computer Science, Volume B: Formal Models and Sematics (B)*, pages 995–1072. North-Holland Pub. Co./MIT Press, 1990.

[15] Matt Fairtlough and Michael Mendler. Propositional lax logic. *Information and Computation*, 137(1):1–33, 1997.

[16] Hector Garcia-Molina and Kenneth Salem. Sagas. In *SIGMOD Conference*, 1987.

[17] Deepak Garg and Martín Abadi. A modal deconstruction of access control logics. In *Proc. FoSSaCS*, pages 216–230, 2008.

[18] Jonathan Gelati, Antonino Rotolo, Giovanni Sartor, and Guido Governatori. Normative autonomy and normative co-ordination: Declarative power, representation, and mandate. *Artificial Intelligence and Law*, 12(1-2):53–81, 2004.

[19] Georgios K. Giannikis and Aspassia Daskalopulu. The representation of e-contracts as default theories. In *New Trends in Applied Artificial Intelligence*, 2007.

[20] Kurt Gödel. Eine interpretation des intuitionistischen aussagenkalkuls. *Ergebnisse eines mathematischen Kolloquiums*, 8:39–40, 1933.

[21] Guido Governatori. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems*, 14(2-3), 2005.

[22] Alain Heuerding, Gerhard Jäger, Stefan Schwendimann, and Michael Seyfried. A logics workbench. *AI Communications*, 9(2):53–58, 1996.

[23] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transactions on Information Systems Security*, 6(1):128–171, 2003.

[24] Joan Moschovakis. Intuitionistic logic. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. 2008.

[25] Luca Padovani. Contract-based discovery and adaptation of web services. In *SFM*, 2009.

[26] The PCL web site. http://www.disi.unitn.it/∼zunino/PCL.

[27] Frank Pfenning. Structural cut elimination - I. intuitionistic and classical logic. *Information and Computation*, 157(1/2):84–141, 2000.

[28] Cristian Prisacariu and Gerardo Schneider. A formal language for electronic contracts. In *Proc. FMOODS*, 2007.

[29] Richard Statman. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Computer Science*, 9:67–72, 1979.

[30] Anne Troelstra and Dirk van Dalen. *Constructivism in Mathematics, vol. 1*. North-Holland, 1988.

A Appendix

A.1 Adapting Pfenning's Notation

For completeness, we show here how to adapt the notation of [27], Appendix 1, to ours. As an example, we adapt the essential case $\wedge R/\wedge L1$. All the other cases are similarly dealt with. In [27] we find the reduction

$$\frac{\dfrac{\overset{N}{\Gamma \to A_1} \quad \overset{N_3}{\Gamma \to A_2}}{\Gamma \to (A_1 \wedge A_2)} \wedge R \otimes \dfrac{\overset{N_4}{\Gamma, (A_1 \wedge A_2), A_1 \to A}}{\Gamma, (A_1 \wedge A_2) \to A} \wedge L1}{\Gamma \to A} \Rightarrow \overset{N_2}{\Gamma \to A}$$

$$\frac{\dfrac{\overset{N}{\Gamma, A_1 \to A_1} \quad \overset{N_3}{\Gamma, A_1 \to A_2}}{\Gamma, A_1 \to (A_1 \wedge A_2)} \wedge R \otimes \overset{N_4}{\Gamma, (A_1 \wedge A_2), A_1 \to A}}{\Gamma, A_1 \to A} \Rightarrow \overset{N_1}{\Gamma, A_1 \to A}$$

$$\overset{N}{\Gamma \to A_1} \otimes \overset{N_1}{\Gamma, A_1 \to A} \Rightarrow \overset{N_2}{\Gamma \to A}$$

which we can read as follows. The sign $\otimes$ in the first line is the reducible cut, the sign $\to$ is our $\vdash$, while N, N_3, N_4 are the subderivations from which we want to construct N_2. This is done in the next lines. First, a recursive call is made in the second line using N, N_3, N_4 in order to obtain N_1. Note that the rightmost derivation w.r.t. $\otimes$ is smaller now. Then, N_1 is used in another recursive call in the third line, together with N, to construct N_2. Here instead the cut formula A_1 is smaller than $(A_1 \wedge A_2)$.

In our notation, we rephrase the above as:

$$\frac{\dfrac{\dfrac{N}{\Gamma \vdash A_1} \quad \dfrac{N_3}{\Gamma \vdash A_2}}{\Gamma \vdash A_1 \wedge A_2} \wedge R \quad \dfrac{\dfrac{N_4}{\Gamma, A_1 \wedge A_2, A_1 \vdash A}}{\Gamma, A_1 \wedge A_2 \vdash A} \wedge L1}{\Gamma \vdash A}\, cut \quad \Longrightarrow$$

$$\frac{\dfrac{N}{\Gamma \vdash A_1} \quad \dfrac{\dfrac{\dfrac{N+}{\Gamma, A_1 \vdash A_1} \quad \dfrac{N_3+}{\Gamma, A_1 \vdash A_2}}{\Gamma, A_1 \vdash A_1 \wedge A_2} \wedge R \quad \dfrac{N_4}{\Gamma, A_1, A_1 \wedge A_2 \vdash A}}{\Gamma, A_1 \vdash A}\, cut_1}{\Gamma \vdash A}\, cut_p$$

A.2 The PCL Tool

To experiment with our logic, we developed a theorem prover for PCL. To prove (or refute) a formula p, it tries to construct a derivation of $\emptyset \vdash p$ using the rules of the sequent calculus. The derivation is constructed in a bottom-up fashion, through an exhaustive search of the proof space. We avoid potential loops (e.g. repeating $weakL$ forever) by pruning the branches which would lead to a derivation having a double occurrence of the same sequent in a root-to-leaf path. In other words, we look for minimal proofs. Lemma 6.14 ensures this is a sound and complete procedure to check for validity in PCL. When a derivation is found, it is provided as output. When the whole proof space is exhausted unsuccessfully, a "no derivation" message is printed.

This simple proof-search technique is inefficient in the worst case. Indeed, PCL is a conservative extension of IPC (Lemma 6.16), so the problem is no easier than checking validity in IPC, which is known to be PSPACE-hard [29]. In all our examples, however, we were able to apply the tool, which required only a few seconds to complete. In our experiments, a depth-first proof search performed worse than a depth-first search, so we adopted the latter in our tool.

The current prototype consists of about 600 lines of Haskell source code. The tool is made available as free software from [26].

Below we show a simple session with the PCL tool:

```
> Pcl '(((a ->> b) /\ (b ->> a)) -> (a /\ b))'
Formula: (((a ->> b) /\ (b ->> a)) -> (a /\ b))
Result

> Pcl '(((p ->> q) /\ (p ->> r)) -> (p ->> (q /\ r)))'
Formula: (((p ->> q) /\ (p ->> r)) -> (p ->> (q /\ r)))
NoResult

> # High verbosity prints search progress and LaTeX derivation
> Pcl --verbosity=4 '((p ->> q) -> ((q -> p) -> q))'
Formula: ((p ->> q) -> ((q -> p) -> q))
1
2
3
4
5
Result (LaTeX follows):
\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{q, (q \imp p), (p \coimp q) \vdash q} id...
```

The last derivation renders as

$$\dfrac{\dfrac{\dfrac{\dfrac{}{q,(q\to p),(p\twoheadrightarrow q)\vdash q}id \quad \dfrac{}{p,q,(q\to p),(p\twoheadrightarrow q)\vdash p}id}{q,(q\to p),(p\twoheadrightarrow q)\vdash p}\to L \quad \dfrac{}{q,(q\to p),(p\twoheadrightarrow q)\vdash q}id}{(q\to p),(p\twoheadrightarrow q)\vdash q}Fix}{(p\twoheadrightarrow q)\vdash((q\to p)\to q)}\to R}{\vdash((p\twoheadrightarrow q)\to((q\to p)\to q))}\to R$$

A.2.1 Auxiliary Results for PCL

Here is the input file we used to prove the negative results in the paper with our PCL tool.

```
#!/bin/sh

Pcl '((p -> q) -> (p ->> q))'

Pcl '((p -> q) -> ((q -> r) -> (p ->> r)))'

Pcl '((p ->> (q \/ r)) -> ((p ->> q) \/ (p ->> r)))'

Pcl '(((p ->> q) /\ (p ->> r)) -> (p ->> (q /\ r)))'

Pcl '((c ->> (c ->> p)) -> (c ->> p))'
```

A.3 Formal Proofs in LWB

Sometimes in our proofs we rely on a theorem prover for checking IPC or S4 tautologies. To this purpose we used the Logics WorkBench tool (LWB) [22]. We provide here the source code we used to perform these checks. The output of the tool, which includes the long, detailed, formal proofs is also available online [26].

A.3.1 Auxiliary Results in IPC

```
# For proof generation, uncomment this.
# generateDetailedProof :== true;

load(ipc);

if generateDetailedProof then set("infolevel", 4);

# Debug.
proc: trace(x)
begin
#print(x);
return x;
end;

# translate a formula
proc: transl(A, phi)
begin
if (phi[0] = AND) then
   return (transl(A, phi[1]) & transl(A, phi[2]));
if (phi[0] = OR) then
   return (transl(A, phi[1]) | transl(A, phi[2]));
if (phi[0] = IMP) then
   return (transl(A, phi[1]) -> transl(A, phi[2]));
if (phi[0] = EQ) then
   return transl(A, (phi[1] -> phi[2]) & (phi[2] -> phi[1]) );
if (phi[0] = SYMBOL) then
   return phi;
if (phi[0] = cimp) then
  begin
  local a, b;
  a := transl(A, phi[1]);
  b := transl(A, phi[2]);
  return A{a/p}{b/q};
  end;
print("ERROR transl!!!!!");
print(phi[0]);
return phi[0];
end;

# Test whether A is sound
proc: test(A)
begin
# axioms: Zero, Fix, PrePost
if not provable(trace(transl(A, cimp(true,true)))) then
   return false;
if not provable(trace(transl(A, cimp(p,p) -> p))) then
   return false;
if not provable(transl(A, (p1 -> p) -> cimp(p,q) -> (q -> q1) -> cimp(p1,q1)))
```

```
   then return false;
return true;
end;

#########################################################

# Known mappings
interp :==
   [ q
   , (q->p) -> q
   , ((q->p) v r) -> q
   , ~~(q->p) -> q
   , ~(q->p) v q
   ] ;

sound :== true;
foreach A in interp do begin
   print("Proving soundness for ",A);
   if not test(A) then begin
      print("unsound: ", A);
      sound :== false;
   end;
end;

if sound then
   print("#### All mappings are sound.");

print("Proving independence");
interp2 :== interp;
indep := true;
while not (interp2 = []) do begin
   A :== pop(interp2);
   foreach B in interp2 do
      if provable(transl(A, cimp(p,q)) <-> transl(B, cimp(p,q))) then begin
         indep :== false;
         print("NOT independent: ", A, " AND ", B);
      end;
end;

if indep then
   print("#### All mappings are independent.");

quit;
```

A.3.2 Auxiliary Results in S4

```
#
# Search for all the sound S4 mappings of contractual implications
# of the form
#     qs[1] ( qs[2] ( qs[3] q -> qs[4] p ) -> qs[5] q )
```

```
# where qs are modalities among identity, box, and diamond.
# The mapping is an extension of the standard IPC-to-S4 mapping.
#

# For proof generation, uncomment this.
# generateDetailedProof :== true;

load(s4);

if generateDetailedProof then set("infolevel", 4);

# apply a modality q (one of i,b,d) to a formula
proc: appMod(q, phi)
begin
if q = i then return phi;
if q = b then return box phi;
if q = d then return dia phi;
print("ERROR appMod");
print(q);
end;

# translate a cimp using the modalities in qs
proc: translCoimpl(qs,p,q)
begin
  return appMod(qs[1],
           ( appMod(qs[2], (appMod(qs[3], q) -> appMod(qs[4], p)))
             ->
             appMod(qs[5], q)
           ));
end;

# translate a formula using the modalities in qs
proc: transl(qs, phi)
begin
if (phi[0] = AND) then
   return (transl(qs, phi[1]) & transl(qs, phi[2]));
if (phi[0] = OR) then
   return (transl(qs, phi[1]) or transl(qs, phi[2]));
if (phi[0] = IMP) then
   return box (transl(qs, phi[1]) -> transl(qs, phi[2]));
if (phi[0] = EQ) then
   return transl(qs, (phi[1] -> phi[2]) & (phi[2] -> phi[1]) );
if (phi[0] = NOT) then
   return box ~ transl(qs, phi[1]);
if (phi[0] = SYMBOL) then
   return box phi;
if (phi[0] = cimp) then
  begin
  local p, q;
  p := transl(qs, phi[1]);
```

```
  q := transl(qs, phi[2]);
  return translCoimpl(qs,p,q);
  end;
print("ERROR transl!!!!");
print(phi[0]);
return phi[0];
end;

# Test whether the modalities qs give rise to a sound cimp
proc: test(qs)
begin
# axioms: Zero, Fix, PrePost
if not provable(transl(qs, cimp(true,true))) then
   return false;
if not provable(transl(qs, cimp(p,p) -> p)) then
   return false;
if not provable(transl(qs, (p1 -> p) -> cimp(p,q) -> (q -> q1) -> cimp(p1,q1)))
   then return false;
return true;
end;


##########################################
# main loop

mods :== [ i , b , d ] ; # modalities: identity, box, dia
sound :== 0 ;
tot :== 0 ;
sol :== [];
foreach q0 in mods do
  foreach q1 in mods do
    foreach q2 in mods do
      foreach q3 in mods do
        foreach q4 in mods do
          begin
          local x;
          x :== [q0,q1,q2,q3,q4];
          tot :== tot + 1;
          if test(x) then
            begin
            sound :== sound + 1;
            sol :== concat(sol,[x]);
            end;
          end;
print("Total formulas: ", tot);
print("Sound formulas: ", num);
#print(sol);


# remove redundant solutions (quotient up to <->)
```

```
sol2 :== [];
foreach s in sol do
   begin
   found :== false;
   foreach s2 in sol2 do
      if provable(transl(s,cimp(p,q)) <-> transl(s2,cimp(p,q))) then
         found :== true;
   if not found then
      sol2 :== concat(sol2, [s]);
   end;

print("Sound formulas up to <->: ", nops(sol2));
print(sol2);
foreach s in sol2 do
   print(transl(s, cimp(p,q)));

print("Completeness check");
foreach s in sol2 do begin
   complete :== true;
   # we try several formulas that are not PCL theorems
   if provable(transl(s, cimp(a,b) <-> b )) then # 1
      complete :== false;
   if provable(transl(s, cimp(a,b) <-> (~(b->a) or b) )) then # 2
      complete :== false;
   if provable(transl(s, cimp(a,b) <-> ((b->a)->b) )) then # 3
      complete :== false;
   if provable(transl(s, cimp(a,b) <-> (~~(b->a)->b) )) then # 4
      complete :== false;
   if not complete then
      print("Formula ", s, " is not complete.");
   else
      print("Formula ", s, " MIGHT be complete.");
end;

quit;
```